AssetCenter™

Version 3.5

Programmer's Reference

March 28, 2000

ITEM ACT-3.5X-EN-00702

AssetCenter and InfraCenter for Workgroups data integrity

AssetCenter and InfraCenter for Workgroups are extremely rich in functionality. This richness relies on a complex database structure: The database contains a large number of tables, fields, links and indexes; certain intermediary tables are not displayed by the graphical interface; certain links, fields and indexes are automatically created, deleted or modified by the software.

Only the interfaces designed for AssetCenter and InfraCenter for Workgroups (graphical interface, APIs, import program, Web interface and gateways) are capable of modifying the database with respect to its integrity. **You must never modify the structure and/or the contents of the database by any means other than those intended for use with the software**; such modifications are highly likely to corrupt the database and bring about symptoms such as involuntary loss or modification of data or links, creation of "ghost" links or records, serious error messages, etc. Alterations to the database resulting from manipulations of this type void the guarantee and technical support provided by Peregrine Systems.

Environments supported by AssetCenter and InfraCenter for Workgroups

The list of environments supported by AssetCenter and InfraCenter for Workgroups can be found in the manual entitled "Installation and Upgrade Guide". Using AssetCenter or InfraCenter for Workgroups in environments other than those for which it is intended is done at the user's risk. Alterations made to the database resulting from using AssetCenter or InfraCenter for Workgroups in environments other than those for which it is intended void the guarantee and technical support provided by Peregrine Systems.

Foreword

This manual aims to:

- Provide users with an exhaustive reference of functions available in the AssetCenter environment.
- Describe the structure of the library of functions, also known as the AssetCenter APIs. This library makes is possible to rapidly develop client applications for AssetCenter databases using development tools such as Visual Basic™ or Access™.

This manual is primarily aimed at readers with solid experience in programming.

Contact Peregrine Systems

World Headquarters

Peregrine Systems, Inc.
3611 Valley Centre Drive
San Diego, CA 92130
USA
Tel: +1 858 481 5000 or 800 638 5231
Fax: +1 858 481 1751
Web: http://www.peregrine.com

Customer support:
Tel: +1 858 794 7402 or 800 960 9998
Fax: +1 858 794 6028
Web: http://support.peregrine.com
E-mail: support@peregrine.com
Open Monday to Friday 5:00 AM to 5:30 PM (PST)

France, Spain, Greece, and Africa (except South Africa)

Peregrine Systems
Tour Franklin - La Défense 8
92042 Paris - La Défense Cedex
France
Tel: +33 (0)1 47 73 11 11
Fax: +33 (0)1 47 73 11 12

Customer support:
Tel: +33 (0) 800 505 100
Fax: +33 (0)1 47 73 11 61
E-mail: frsupport@peregrine.fr
Open Monday to Friday 8:00 AM to 6:00 PM (local time)

Germany and Eastern Europe

Peregrine Systems GmbH
Bürohaus ATRICOM
Lyoner Strasse 15

60528 Frankfurt
Germany
Tel: +49 (0)(69) 6 77 34-0
Fax: +49 (0)(69) 66 80 26-26

Customer support:
Tel: 0800 2773823
Fax: +49 (0)(69) 66 80 26-26
E-mail: psc@peregrine.de
Open Monday to Friday 8:00 AM to 5:00 PM (local time)

United Kingdom

Peregrine Systems, Ltd.
Ambassador House
Paradise Road
Richmond
Surrey TW9 1SQ
UK
Tel: +44 (0)181 332 9666
Fax: +44 (0)181 332 9533

Customer support:
Tel: +44 (0)181 334 5844 or 0800 834 7700
Fax: +44 (0)181 334 5890
E-mail: uksupport@peregrine.com
Open Monday to Friday 8:00 AM to 6:00 PM (local time)

Denmark, Norway, Finland and Iceland

Peregrine Systems A/S
Naverland 2, 12th fl.
DK-2600 Glostrup
Denmark
Tel: +45 43 46 76 76
Fax : +45 43 46 76 77

Customer support:
Tel: +45 77 31 77 76
Fax: +45 43 46 76 77
E-mail: support.nordic@peregrine.com
Open Monday to Friday 8:30 AM to 4:00 PM (local time)

The Netherlands, Belgium, and Luxembourg

Peregrine Systems BV
Botnische Golf 9a
Postbus 244
3440 AE Woerden
The Netherlands
Tel: +31 (0) 348 43 7070
Fax: +31 (0) 348 43 7080

Customer support:
Tel: 0800 0230889 (The Netherlands)
or 0800 74747575 (Belgium and Luxembourg)
Fax: +31 (0) 348 43 7080
E-mail: benelux.support@peregrine.com
Open Monday to Friday 8:00 AM to 6:00 PM (local time)

Singapore

Peregrine Systems Pte.Ltd
#03-16
CINTECH III
77 Science Park Drive
Singapore Science Park
118256
Singapore
Tel: +65 778 5505
Fax: +65 777 3033

Italy

Peregrine Systems, S.r.l.
Via Monte di Pietà, 21
I-20121 Milano
Italy
Tel: +39 (02) 6556931
Fax: +39 (02) 65569390

Sweden

Peregrine Systems AB
Frösundaviks Allé 15, 4th floor
S-169 70 Solna
Sweden

Tel: +46 (0)8-655 36 04
Fax : +46 (0)8-655 26 10

Customer support:
Tel: +45 77 31 77 76
Fax: +45 43 46 76 77
E-mail: nordic@peregrine.com
Open Monday to Friday 8:30 AM to 4:30 PM (local time)

Japan

Peregrine Systems K.K.
Level 32, Shinjuku Nomura Building
1-26-2 Nishi-shinjuku, Shinjuku-ku
Tokyo 163-0532
Japan

Tel: +81 (3) 5322-1350
Fax: +81 (3) 5322-1352

Customer support:
Tel: +81 (3) 5322 1350
Fax: +81 (3) 5322 1352
E-mail: glipper@peregrine.com

Conventions

The following notation is used for commands:

`[ ]`	Square brackets denote an optional parameter. Do not type them in your command. Exception: In BASIC scripts, square brackets are used to denote the data access path and must be included in the script: **`[Link.Link.Field]`**
`< >`	Brackets denote a parameter in plain language. Do not type them. Substitute the text with the appropriate information.
`{ }`	Curly brackets denote a series of parameters. Only one of these parameters may be used. Do not type these curly brackets in your command.
`|`	A pipe is used to separate a series of parameters contained within curly brackets.
*	An asterisk added to the right of square brackets means that the formula shown can be repeated several times.

The following text formats have given meanings:

`Fixed width characters`	DOS command.
`Example`	Example of code or command.
...	Code or command omitted.
Object name	The names of fields, tabs, menus and files are shown in bold.
Note	Important note.

Send us your comments

We want to deliver the most accurate documentation possible.

Any comments would be greatly appreciated.

Send any remarks to **documentation@peregrine.com**.

Table of contents

Chapter 1 - Introduction **1**
- Classification of functions 1
  - Families of functions 1
  - Field of application 2
  - Functional domains 3
- Conventions 3
  - Notation 4
  - Format of "Date+Time" constants in scripts 5
  - Format of "Duration" type constants in scripts 5
- Definitions 6
  - Definition of a function 6
  - Definition of the "CurrentUser" virtual link 7
  - Definition of a handle 7
  - Definition of an error code 7
- Function typing and parameters 8
  - List of types 8
  - Type of a function 9
  - Type of a parameter 9

Chapter 2 - Using APIs **11**
- Introduction 11
  - Warning 11
  - Installation 12
- Methodology 12
- Concepts and examples 13
  - Concepts 13
  - Handling dates 13
  - First example 14
  - Second example 15

Chapter 3 - Alphabetical Reference **17**
- Abs() 17
- AmActionDde() 19
- AmActionExec() 20
- AmActionMail() 22
- AmActionPrint() 24
- AmAddAllPOLinesToInv() 26

AmAddEstimLinesToPO() 27
AmAddEstimLineToPO() 28
AmAddPOLineToInv() 30
AmAddReqLinesToEstim() 31
AmAddReqLinesToPO() 33
AmAddReqLineToEstim() 34
AmAddReqLineToPO() 36
AmApproveRequest() 37
AmBusinessSecondsInDay() 38
AmCalcConsolidatedFeature() 40
AmCalcDepr() 41
AmCleanup() 43
AmClearLastError() 44
AmCloseAllChildren() 45
AmCloseConnection() 46
AmCommit() 47
AmComputeAllLicAndInstallCounts() 48
AmComputeLicAndInstallCounts() 49
AmConvertCurrency() 51
AmConvertDateBasicToUnix() 53
AmConvertDateIntlToUnix() 54
AmConvertDateStringToUnix() 55
AmConvertDateUnixToBasic() 57
AmConvertDateUnixToIntl() 58
AmConvertDateUnixToString() 59
AmConvertDoubleToString() 61
AmConvertMonetaryToString() 62
AmConvertStringToDouble() 63
AmConvertStringToMonetary() 65
AmCounter() 66
AmCreateDelivFromPO() 68
AmCreateEstimFromReq() 69
AmCreateEstimsFromAllReqLines() 70
AmCreateInvFromPO() 72
AmCreateLink() 73
AmCreatePOFromEstim() 75
AmCreatePOFromReq() 76
AmCreatePOsFromAllReqLines() 77
AmCreateRecord() 79
AmCurrentDate() 80
AmCurrentLanguage() 81
AmCurrentServerDate() 82
AmDateAdd() 84
AmDateAddLogical() 85
AmDateDiff() 87
AmDbGetDate() 88
AmDbGetDouble() 90
AmDbGetList() 91
AmDbGetListEx() 93

AmDbGetLong() 94
AmDbGetPk() 96
AmDbGetString() 97
AmDbGetStringEx() 99
AmDeadLine() 100
AmDecrementLogLevel() 102
AmDefaultCurrency() 103
AmDeleteLink() 104
AmDeleteRecord() 105
AmEndOfNthBusinessDay() 107
AmEnumValList() 108
AmExecTransition() 110
AmExecuteActionById() 111
AmExecuteActionByName() 113
AmGenSqlName() 114
AmGetComputeString() 116
AmGetFeat() 117
AmGetFeatCount() 118
AmGetField() 119
AmGetFieldBlobLength() 121
AmGetFieldBlobValue() 122
AmGetFieldCount() 123
AmGetFieldDateValue() 125
AmGetFieldDescription() 126
AmGetFieldDoubleValue() 127
AmGetFieldFormat() 129
AmGetFieldFormatFromName() 131
AmGetFieldFromName() 132
AmGetFieldLabel() 133
AmGetFieldLabelFromName() 135
AmGetFieldLongValue() 136
AmGetFieldName() 138
AmGetFieldRights() 139
AmGetFieldSize() 141
AmGetFieldSqlName() 142
AmGetFieldStrValue() 143
AmGetFieldType() 145
AmGetFieldUserType() 147
AmGetForeignKey() 149
AmGetIndex() 150
AmGetIndexCount() 151
AmGetIndexField() 153
AmGetIndexFieldCount() 153
AmGetIndexFlags() 154
AmGetIndexName() 155
AmGetLink() 156
AmGetLinkCardinality() 158
AmGetLinkCount() 159
AmGetLinkDstField() 160

AmGetLinkFeatureValue() 161
AmGetLinkFromName() 163
AmGetLinkType() 164
AmGetMainField() 165
AmGetRecordFromMainId() 166
AmGetRecordHandle() 168
AmGetRecordId() 169
AmGetReverseLink() 170
AmGetSelfFromMainId() 171
AmGetSourceTable() 173
AmGetTable() 174
AmGetTableCount() 175
AmGetTableDescription() 176
AmGetTableFromName() 178
AmGetTableLabel() 179
AmGetTableName() 180
AmGetTableRights() 182
AmGetTableSqlName() 183
AmGetTargetTable() 184
AmGetTypedLinkField() 186
AmGetVersion() 187
AmHasAdminPrivilege() 188
AmIncrementLogLevel() 189
AmInsertRecord() 191
AmIsConnected() 192
AmIsFieldForeignKey() 193
AmIsFieldIndexed() 194
AmIsFieldPrimaryKey() 196
AmIsLink() 197
AmIsTypedLink() 198
AmLastError() 199
AmLastErrorMsg() 201
AmListToString() 202
AmLog() 203
AmLoginId() 205
AmLoginName() 206
AmMsgBox() 207
AmOpenConnection() 209
AmPagePath() 210
AmProgress() 211
AmQueryCreate() 213
AmQueryExec() 214
AmQueryGet() 215
AmQueryNext() 216
AmQueryStartTable() 218
AmQueryStop() 219
AmReceiveAllPOLines() 220
AmReceivePOLine() 222
AmRefreshAllCaches() 223

AmRefreshProperty() 224
AmRefuseRequest() 226
AmReleaseHandle() 227
AmRgbColor() 228
AmRollback() 230
AmSeCreateDefVal() 231
AmSetFieldDateValue() 233
AmSetFieldDoubleValue() 234
AmSetFieldLongValue() 236
AmSetFieldStrValue() 237
AmSetLinkFeatureValue() 238
AmSetProperty() 240
AmSetUseUserName() 241
AmStartTransaction() 242
AmStartup() 243
AmTableDesc() 244
AmTaxRate() 246
AmUpdateDetail() 247
AmUpdateLoginSlot() 249
AmUpdateRecord() 250
AmwAdminLogout() 251
AmwApproveRequest() 252
AmwCloseSlave() 254
AmwCurrentCnx() 256
AmwCurrentUserId() 257
AmwErrorMsg() 258
AmwExist() 259
AmwFetchQuery() 261
AmwGetEnv() 263
AmwGetTpl() 264
AmwInclude() 265
AmWizChain() 267
AmwLoadFile() 268
AmwLogin() 269
AmwLogout() 271
AmwNewRecord() 272
AmWorkTimeSpanBetween() 274
AmwOutputIf() 276
AmwOutputIfNot() 278
AmwPlusToSpace() 279
AmwPrintDebug() 281
AmwQStrDelKey() 282
AmwQStrSetKey() 284
AmwRefuseRequest() 285
AmwSetEnv() 287
AmwSetTpl() 288
AmwSetWebAdminPwd() 290
AmwSetWebTimeout() 291
AmwSlaveList() 293

AmwSpaceToPlus() 294
AmwStrReplace() 295
AmwUpdateRecord() 297
AmwVars() 299
AppendOperand() 300
ApplyNewVals() 301
Asc() 303
Atn() 304
BasicToLocalDate() 306
BasicToLocalTime() 307
BasicToLocalTimeStamp() 308
Beep() 309
CDbl() 310
ChDir() 312
ChDrive() 313
Chr() 314
CInt() 316
CLng() 317
Cos() 319
CountOccurences() 320
CountValues() 321
CSng() 323
CStr() 324
CurDir() 325
CVar() 326
Date() 327
DateSerial() 329
DateValue() 330
Day() 332
EscapeSeparators() 333
Exp() 334
ExtractValue() 336
FileCopy() 337
FileDateTime() 338
FileLen() 340
Fix() 341
Format() 342
FormatDate() 343
FormatResString() 345
FormatString() 346
FV() 348
GetListItem() 350
Hex() 351
Hour() 352
InStr() 354
Int() 355
IPMT() 356
Kill() 358
LCase() 359

Left() 361
LeftPart() 363
LeftPartFromRight() 365
Len() 367
LocalToBasicDate() 368
LocalToBasicTime() 370
LocalToBasicTimeStamp() 371
Log() 372
LTrim() 373
MakeInvertBool() 375
Mid() 376
Minute() 378
MkDir() 380
Month() 381
Name() 382
Now() 383
NPER() 384
Oct() 386
ParseDMYDate() 387
ParseMDYDate() 389
ParseYMDDate() 390
PMT() 391
PPMT() 393
PV() 395
Randomize() 397
RATE() 398
RemoveRows() 400
Replace() 402
Right() 404
RightPart() 406
RightPartFromLeft() 408
RmDir() 409
Rnd() 410
RTrim() 412
Second() 414
SetSubList() 415
Sgn() 417
Shell() 418
Sin() 420
Space() 421
Sqr() 422
Str() 424
StrComp() 425
String() 426
SubList() 427
Tan() 429
Time() 431
Timer() 432
TimeSerial() 433

TimeValue() 435
Trim() 436
UCase() 438
UnEscapeSeparators() 440
Union() 441
Val() 443
WeekDay() 444
Year() 445

Chapter 1 - Introduction

This section contains information on the following:

- *Classification of functions*
- *Conventions*
- *Definitions*
- *Function typing and parameters*

Classification of functions

Functions can be classified according to three different levels:

- *Families of functions*
- *Field of application*
- *Functional domains*

Families of functions

Functions in the AssetCenter environment can be organized into three main families:

- Functions recognized by AssetCenter: These are essentially functions that can be used in the scriptable parts (in Basic) of the software.
- Functions recognized by the AssetCenter APIs: These functions can be called by external tools or be a program written in a high-level language.
- Functions recognized by the AssetCenter Web API library: These functions can be used in the script sections of HTML templates.

These three families of functions are not mutually exclusive. For example, certain AssetCenter API functions can be used in the Basic scripts in the software. Such a function, originating from the AssetCenter APIs is said to be "exposed" in AssetCenter's internal Basic scripts. The syntax of such a function may change but its behavior remains the same.

Field of application

The functions described in this document can be used in at least one of the following contexts:

- All script windows in AssetCenter. These are known as "built-in" functions.
- AssetCenter APIs
- Field or link configuration script (Configure object popup menu item or AssetCenter Database Administrator) and by extension Calculation script (Nom SQL : memScript) of a calculated field:
  - Default value.
  - Mandatory nature.
  - Historization.
  - Read-only nature.
- Feature parameter configuration script:
  - Default value.
  - Availability.
  - Force display.
  - Mandatory nature.
  - Historization.
- Script type actions:
  - Script defined in the Action script (Nom SQL : Script) field of a Script action.
- AssetCenter wizards:
  - "FINISH.DO" script of a wizard.
  - Value definition scripts for the properties of nodes.
- AssetCenter Web API librairies.

Functional domains

Each function is associated with a functional domain. This functional domains describes the nature of operations carried out by the function. The different functional domains are listed below::

- Connection, handling
- Table, query and rrecord object handling
- Record and query object handling
- Field and link handling
- Table handling
- Query handling
- Record handling
- Transaction handling
- Direct data access
- Helpdesk Management
- Software Management
- Procurement Management
- Conversion
- Finance
- Actions
- Date and time calculations
- Calculations
- String handling
- File management
- Error handling
- Wizard functions
- Web functions
- Miscellaneous

Conventions

This chapter describes:

- *Notation*
- *Format of "Date+Time" constants in scripts*
- *Format of "Duration" type constants in scripts*

Notation

The following notation is used in the examples in this manual:

[]	Square brackets denote an optional parameter. Do not type these brackets in your command. Exception: In Basic scripts, when the square brackets denote the path to data in the database with the form: `[Link.Link.Field]` These brackets must appear in the script.
<>	Angle brackets denote a parameter in plain language. Do not type these brackets. Substitute the text with the appropriate information.
{}	Curly brackets denote a series of parameters. Only one of these parameters may be used. Do not type these curly brackets in your command.
|	A pipe is used to separate a series of possible parameters contained within curly brackets.
*	The asterisk added to the right of the curly brackets indicates that the formula may be repeated several times.

The following text styles have specific meanings:

`Fixed width characters`	DOS command, function parameter are data formatting.
`Example`	Example of code or command.
...	Code or command omitted.
Object name	The names of fields, tabs, menus and files are shown in bold.

Note: Note	Important note.

Format of "Date+Time" constants in scripts

Dates referenced in scripts are expressed in international format, independently of the display options specified gby the user:

```
yyyy/mm/dd hh:mm:ss
```

Example:

```
RetVal="1998/07/12 13:05:00"
```

Note: The hypen ("-") can also be used as a date separator.

"Basic" and "Unix" date

Dates are expressed differently in Basic and under Unix:

- In Basic, a date can be expressed in international format, or as a floating point number ("Double" type). In this case, the integer part of the number represents the number of days elapsed since 12/30/1899 at midnight, the decimal part represents the fraction of the current date (The number of seconds elapsed since the start of the day divided by 86400).
- Under Unix, dates are expressed as a long integer ("Long" type) that represents the number of seconds elapsed since 01/01/1870 at midnight, independent of time zones (UTC time).

Note: When they are called by an external program, all date processing function expect and return GMT dates.

Format of "Duration" type constants in scripts

In scripts, durations are stored and expressed in seconds. E.g. to set the default value for a "Duration" type field to 3 days, use the following script:

```
RetVal=259200
```

Likewise, functions that calculate durations, such as the `AmWorkTimeSpanBetween` function, return a number of seconds.

Note: In financial calculations, AssetCenter takes into account the most common simplifications used. In this case alone, a year is considered as being 12 months and 1 month as 30 days (thus: 1 year = 360 days).

Definitions

This chapter groups together the definitions of several essential terms.

You will find the following definitions:

- *Definition of a function*
- *Definition of the "CurrentUser" virtual link*
- *Definition of a handle*
- *Definition of an error code*

Definition of a function

A function is a program that performs operations and returns a value to the user. This value is called the "return value" or "return code".

Here is an example of the syntax used to call an internal AssetCenter function:

```
AmConvertCurrency(strSrcName As String, strDstName As String, dVal As
Double) As Double
```

Here is the syntax of the same function via the AssetCenter APIs:

```
double AmConvertCurrency(long hApiCnxBase,long ltm, const char
*pszSrcName, const char *pszDstName,double dVal)
```

Definition of the "CurrentUser" virtual link

Definition

"CurrentUser" can be considered as a link starting in all tables and pointing to the record in the table of departments and employees corresponding to the current user.

- In the "CurrentUser" format, it points to the record corresponding to the current user, and returns the user's ID number.
- In the "CurrentUser.Field" format, it returns the value of the field for the current user.

Note: This virtual link is not displayed in the list of fields and links; therefore it is not directly accessible in AssetCenter's internal script builder. You must enter this expression manually.

Equivalencies

The `AmLoginName` and `AmLoginId functions, which return the current user's "Name" (SQL name: Name) and ID (SQL name: lPersId), respectively, may be considered as functions derived from "CurrentUser". In effect, the following are equivalent:`

- AmLoginName()=[CurrentUser.Name]
- AmLoginId()=[CurrentUser.lPersId]

Definition of a handle

A handle represents a unique identifier on an object. In the context of AssetCenter, this object can be a field, link, query, record, table or a connection. Handles are 32-bit integers ("Long" type).

Note: The NULL value is not a valid handle.

Definition of an error code

When a function fails, it returns an error code.

This error code and associated message can be recovered using the `AmLastError` and `AmLastErrorMsg` functions respectively. It can be cleared using the `AmClearLastError` function.

Note: Any new function calls clear the error code and previous message.

You can trigger an error message intentionally using the `Err.Raise` function. Its syntax is as follows:

```
Err.Raise (<Error number>, <Error message>)
```

Note: When creation or modification of a record is invalidated by the value of the "Validity" field for the table concerned, it is good practice to display an error message using the `Err.Raise` function in order to warn the user. If you do not do this, the user will not necessarily understand why it is not possible to modify or create a record.

Function typing and parameters

This chapter contains information on the following:

- *List of types*
- *Type of a function*
- *Type of a parameter*

List of types

The following table summarizes the various types available for a function or a parameter:

Type	Description
Integer	Integer from -32,768 to +32,767.
Long	Integer from -2,147,483,647 to +2,147,483,646.
Double	8 byte floating-point number.
String	Text in which all characters are allowed.
Date	Date or Date+Time.
Variant	Generic type that can represent any type.

Type of a function

The type of a function corresponds to the type of the value returned by the function. We recommend paying close attention to this piece of information since it can be at the origin of compilation and runtime errors in your programs.

For example, you cannot use a function returning a certain typed value in the definition of the default value of a differently typed field. Try, for example, assigning this default value script to any "Date" or "Date+Time" type field:

```
RetVal=AmLoginName()
```

The `AmLoginName` function returns the name of the connected user in the form of a character string ("String" type). The type of this return value is therefore incompatible with "Date" type fields and AssetCenter displays an error message.

Type of a parameter

The parameters that can be used in functions also have a type that must be respected in order for the proper execution of the function. In the syntax of functions, parameters are prefixed according to their type. To avoid any possible confusion, the prefixes used in this reference differ according to the syntax (API or Basic) of the function. The following table resumes the equivalencies between the prefixes used in the API syntax and the Basic syntax:

Type	Prefix used in the API syntax	Prefix used in the Basic syntax
Integer	"i"	"i"
Long	"h" for a handle or "l" for a number	"l"
Double	"d"	"d"
String	"char*psz"	"str"
Date	"ltm"	"dt"
Variant	"v"	"v"

Chapter 2 - Using APIs

This chapter provides an introduction to using the APIs. The following topics are covered:

- *Introduction*
- *Methodology*
- *Concepts and examples*

Introduction

The APIs are provided as a 32-bit DLL useable under Windows 95/ 98 or Windows NT.

It had been tested successfully under the following environments:

- Visual Basic 4.0, 5.0 and 6.0,
- Visual C++ 4.0, 5.0 and 6.0,
- All Microsoft™ products using VBA (Visual Basic for Applications)

Note: The APIs should be compatible with all tools authorizing the user of third-party DLLs.

Warning

Before using AssetCenter API, the user should be familiar with the terminology used in the AssetCenter conceptual model. In particular, a minimal knowledge of the database structure is required.

Information on the structure of the database can be found in the manual entitled "Reference guide: Administration and advanced use", chapter "Structure of the AssetCenter database" and in the "database.txt" and

"tables.txt" files, which can be found in the "infos" sub-folder of the AssetCenter installation folder.

Installation

Before using the DLL, it is highly recommended to install a fully functional version of AssetCenter. In this way, a quick test can be done to check that databases can be correctly accessed from a particular computer. AssetCenter can be used as an easy way to create or configure database connections. The DLL uses the same database layers and the same configuration information as AssetCenter to access datasources, so problems can often be investigated from AssetCenter.

The typical steps for setting up a development environment should be:

- Installing a 32-bit version of AssetCenter with the AssetCenter API package.
- Use AssetCenter to configure the datasource, and try to open a database.
- Use your development environment to call AssetCenter DLL functions.

It is highly recommended using a demo database or any non critical source of data for which manipulation errors are not critical.

Methodology

A typical sequence of operations would be:

1. Create query using an AQL sentence:

```
SELECT AssetTag, User.Name, Supervisor.Name FROM amAsset
```

Note: Using AssetCenter Export is a good solution for easy creation of AQL queries.

2. Navigate through the query result set and retrieve any useful handles on specific items.

3. Use retrieved handles to update record information.

4. Commit (accept) or rollback (cancel) the whole transaction.

Concepts and examples

This section contains information on the following:

- *Concepts*
- *Handling dates*
- *First example*
- *Second example*

Concepts

AssetCenter is built around an object oriented design and the DLL maintains this structural view. To accommodate the limitation of Windows DLLs which imposes the use of a flat "C like API", AssetCenter DLL uses handles (32-bit integers) to identify every user-created object. This approach has the advantage of allowing non-object oriented languages to access the AssetCenter object model.

Before doing anything else, your program must call `AmStartUp` in order to initialize the AssetCenter DLL. Your program must also terminate by calling the `AmCleanUp` function.

Before accessing AssetCenter objects, a connection should be established between the user and AssetCenter database. This connection is identified by a "handle" on a "connection" object (this handle is then used in all the API functions that interact with the database. It corresponds to the parameter *`hApiCnxBase`*. This object can then be used to create queries and gain access to records.

Note: All database objects are linked to a connection so information about user privileges can be checked.

The first step is to open a connection using a valid datasource name and a valid login/password combination.

Warning: When you connect to the AssetCenter database via the APIs, a connection slot is used.

Handling dates

When reading dates, you have the choice of the following two functions for "Date" and "Date+Time" type fields:

- `AmGetFieldLongValue` which returns the date as a Unix "Long" (UTC). We recommend using this function for calculations involving dates.
- `AmGetFieldStrValue` which returns the date as a string in the same format as the Windows Control Panel. This date takes time zones into account. We recommend using this function when you need to display a date.

First example

An example in C of connection creation to the demo database might be:

```
long lCnx
lCnx = AmOpenConnection("ACDemo300ENG","Admin","");
```

`lCnx` is a connection handle that can be used to identify the newly created connection.

Now, this connection can be used to create queries and access the database. A 'C' example of navigating through specific assets can be:

```
#include “apiproto.h”
#define SZ_MODEL_LEN 200
long lCnx ;
long lQuery ;
long lStatus ;  /* to store error code */
char szModel[SZ_MODEL_LEN] ;
/* dll initialization */
AmStartup();
/* Open a connection */
lCnx = AmOpenConnection(“ACDemo300Eng”,”Admin” ,””) ;
if( lCnx != 0 )
{
  /* Creation of a query object */
  lQuery = AmQueryCreate (lCnx)
  if( lQuery != 0 )
  {
    /* Construction of  the result set : all assets from Compaq*/
  lStatus = AmQueryExec(lQuery, “select AssetTag where brand =
‘Compaq'”)
    /* Navigates through the result set */
    while( !lStatus )
    {
        /* Read the first field (AssetTag) of the current item in the
query */
        lStatus = AmGetFieldStrValue(lQuery,0,szModel,SZ_MODEL_LEN-1);
        if( lStatus == 0 )
        {
        printf(‘ Compaq AssetTag=%s\n',szModel);
        lStatus = AmQueryNext(lQuery);
      }
    }
    /* clean things up */
```

```
    AmReleaseHandle(lQuery);
  }
  AmCloseConnection(lCnx);
}
AmCleanup();
```

Second example

Queries are used to locate objects in the database, when updates should be done, a handle on the target record should be obtained from the query. Specific API can then be used to manipulate records.

The next example shows how to modify a field from a specific record:

```
/* Handles for objects */
long lCnx ;
long lQuery ;
long lStatus ;
long lRecord ;
AmStartup();
lCnx = AmOpenConnection("ACDemo300Eng","Admin" ,"") ;
/* Creation of a query object attached to lCnx */
lQuery = AmQueryCreate(lCnx);
/* Mark the starting point of the current transaction */
AmStartTransaction(lCnx);
/* Use a query that matches a single object */
lStatus = AmQueryGet(lQuery, "select model, AssetId where brand =
'Compaq' and barcode='34234'") ;
/* Get a record handle to the matching object */
lRecord = AmGetRecordHandle(lQuery) ;
/* Change the field Field1 with new value spam */
lStatus = AmSetFieldStrValue(lRecord, "Field1", "Spam");
/* Update the change for the current session */
lStatus = AmUpdateRecord(lrecord);
/* Commit all modifications to the database */
lStatus = AmCommit(lCnx) ;
/* you can release here query and record objects */
/* but closing connection will do it */
/* Close the connection to the database */
AmCloseConnection(lCnx);
AmCleanup();
```

This example shows how to get a unique identifier (handle) on a record using the query mechanism. In this sample code, the query is used to locate a single item, but it is also possible to use `AmQueryExec` to get a set of records and get a record handle for one or several records during browsing.

Note: In order to simplify this example, not all error codes are dealt with.

Chapter 3 - Alphabetical Reference

Abs()

Internal BASIC syntax

```
Function Abs(dValue As Double) As Double
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Returns the absolute value of a number.

Input parameters

- *dValue*: Number for which you want to know the absolute value.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Example

```
Sub Main
    Dim Msg, X, Y
    X = InputBox("Enter a Number:")
    Y = Abs(X)
    Msg = "The number you entered is "  & X
    Msg = Msg + ". The Absolute value of " & X & " is " & Y
    Print Msg 'Display Message.
End Sub
```

AmActionDde()

API syntax

```
long AmActionDde(char *strService, char *strTopic, char *strCommand,
char *strFileName, char *strDirectory, char *strParameters, char
*strTable, long lRecordId);
```

Internal BASIC syntax

```
Function AmActionDde(strService As String, strTopic As String,
strCommand As String, strFileName As String, strDirectory As String,
strParameters As String, strTable As String, lRecordId As Long) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	X
Wizard script	
FINISH.DO script of a wizard	X
AssetCenter Web APIs	X

Description

This function sends a DDE request to a DDE server application. Using this function, AssetCenter can control another application using a DDE link.

Input parameters

- *strService*: This parameter contains the name of the DDE service provided by the executable you want to call. Refer to the documentation of the executable to obtain the list of DDE services it provides.
- *strTopic*: This parameter contains the topic (i.e. the context) in which a DDE action must be executed.
- *strCommand*: This parameter contains the commands the external application must execute. You must follow the syntax defined by the external application.
- *strFileName*: If the service is not resident in memory, you must load it by specifying in this parameter the name of the executable (or the name of any file associated with an executable using the Windows File Manager) which activates the service.
- *strDirectory*: This parameter contains the path for the file defined in *strFileName*.
- *strParameters*: This parameter contains the various parameters to pass to the executable which activates the service when it is launched.
- *strTable*: Optional parameter used when the action is contextual. It indicates the SQL name of the table containing the record to which the action applies.
- *lRecordId*: Optional parameter used when the action is contextual. It indicates the identifier of the record to which the action applies.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmActionExec()

API syntax

```
long AmActionExec(char *strFileName, char *strDirectory, char
*strParameters, char *strTable, long lRecordId);
```

Internal BASIC syntax

```
Function AmActionExec(strFileName As String, strDirectory As String,
strParameters As String, strTable As String, lRecordId As Long) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	X
Wizard script	
FINISH.DO script of a wizard	X
AssetCenter Web APIs	X

Description

This function launches an ".exe", ".com", ".bat", ".pif" application. You can also refer to any type of document, as long as the document extension is associated with an executable via the Windows File Manager. This function is equivalent to an action of "Executable" type.

Input parameters

- `strFileName`: This parameter contains the name of the executable, or of any document (associated with an executable via the File Manager).
- `strDirectory`: This parameter contains the path for the file specified in the `strFileName` parameter..
- `strParameters`: This optional parameter contains the various parameters to be provided to the executable when it is launched.

- *strTable*: Optional parameter used when the action is contextual. It indicates the SQL name of the table containing the record to which the action applies.
- *lRecordId*: Optional parameter used when the action is contextual. It indicates the identifier of the record to which the action applies.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

Example

This example executes the Windows NT explorer (situated in the "WinNT" folder of the "C" drive):

```
AmActionExec(explorer.exe, c:\winnt\, )
```

AmActionMail()

API syntax

```
long AmActionMail(char *strTo, char *strCc, char *strCcc, char
*strSubject, char *strMessage, long iPriority, long bAcknowledge, char
*strRefObject, char *strTable, long lRecordId);
```

Internal BASIC syntax

```
Function AmActionMail(strTo As String, strCc As String, strCcc As
String, strSubject As String, strMessage As String, iPriority As Long,
bAcknowledge As Long, strRefObject As String, strTable As String,
lRecordId As Long) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	X
Wizard script	
FINISH.DO script of a wizard	X
AssetCenter Web APIs	X

Description

This function sends a message via one of the messaging systems managed by AssetCenter:

- Internal messaging system.
- External messaging system based on the VIM standard (Lotus Notes, etc.)
- External messaging system based on the MAPI standard (Microsoft Exchange, Microsoft Outlook, etc.)
- External messaging system based on the SMTP standard (Internet standard)

Input parameters

- `strTo`: This parameter contains the list of addresses for the message recipients. The semi-colon is used as a separator.
- `strCc`: This parameter contains the list of addresses for people who are copied in the message. The semi-colon is used as a separator.
- `strCcc`: This parameter contains the list of addresses for people who receive a blind copy of the message (they do not appear in the list of recipients). The semi-colon is used as a separator.
- `strSubject`: This parameter contains the message subject.

- *strMessage*: This parameter contains the message body.
- *iPriority*: This parameter defines the priority for sending the message:
  - 0: Low priority
  - 1: Normal priority.
  - 2: High priority.
- *bAcknowledge*: This parameter indicates whether the message sender will receive an acknowledgement:
  - 0: the sender does not receive an acknowledgement.
  - 1: the sender does receive an acknowledgement.
- *strRefObject*: This parameter is only used for messages sent via the AssetCenter internal messaging system. It gives direct access to the object that triggered the sending of the message.
- *strTable*: Optional parameter used when the action is contextual. It indicates the SQL name of the table containing the record to which the action applies.
- *lRecordId*: Optional parameter used when the action is contextual. It indicates the identifier of the record to which the action applies.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmActionPrint()

API syntax

```
long AmActionPrint(long lReportId, long lRecordId);
```

Internal BASIC syntax

```
Function AmActionPrint(lReportId As Long, lRecordId As Long) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	X
Wizard script	
FINISH.DO script of a wizard	X
AssetCenter Web APIs	X

Description

This function prints a report on a given record in the database.

Input parameters

- `lReportId`: This parameter contains the identifier of the report to be printed.
- `lRecordId`: This parameter contains the identifier of the record concerned by the report. By default, this parameter is set to "0".

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmAddAllPOLinesToInv()

API syntax

```
long AmAddAllPOLinesToInv(long hApiCnxBase, long lPOrdId, long lInvId);
```

Internal BASIC syntax

```
Function AmAddAllPOLinesToInv(lPOrdId As Long, lInvId As Long) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	X
Wizard script	
FINISH.DO script of a wizard	X
AssetCenter Web APIs	X

Description

This function adds a purchase order in full to an existing supplier invoice.

Input parameters

- `lPOrdId`: This parameter contains the identifier of the order to be added to the supplier invoice.
- `lInvId`: This parameter contains the identifier of the supplier invoice to which the purchase order is added.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmAddEstimLinesToPO()

API syntax

```
long AmAddEstimLinesToPO(long hApiCnxBase, long lEstimId, long lPOrdId,
long bMergeLines);
```

Internal BASIC syntax

```
Function AmAddEstimLinesToPO(lEstimId As Long, lPOrdId As Long,
bMergeLines As Long) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	X

	Available
Wizard script	
FINISH.DO script of a wizard	X
AssetCenter Web APIs	X

Description

This function adds all estimate lines of an estimates to an existing order.

Input parameters

- `lEstimId`: This parameter contains the identifier of the estimate to be added to the purchase order.
- `lPOrdId`: This parameter contains the identifier of the purchase order to which the all the estimate lines of the estimate are added.
- `bMergeLines`: This parameter allows you to specify if identical request lines are to be combined (`bMergeLines`=0) to give one single line. The quantities given for the lines to be combined are added together and a single line is created.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmAddEstimLineToPO()

API syntax

```
long AmAddEstimLineToPO(long hApiCnxBase, long lEstimLineId, long
lPOrdId, long bMergeLines);
```

Internal BASIC syntax

```
Function AmAddEstimLineToPO(lEstimLineId As Long, lPOrdId As Long,
bMergeLines As Long) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	X
Wizard script	
FINISH.DO script of a wizard	X
AssetCenter Web APIs	X

Description

This function adds an estimate line to an existing purchase order.

Input parameters

- `lEstimLineId`: This parameter contains the identifier of the estimate line to be added to the purchase order.
- `lPOrdId`: This parameter contains the identifier of the purchase order to which the estimate line is added.
- `bMergeLines`: This parameter allows you to specify if identical request lines are to be combined (`bMergeLines`=0) to given one single line. The quantities given for the lines to be combined are added together and a single line is created.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmAddPOLineToInv()

API syntax

```
long AmAddPOLineToInv(long hApiCnxBase, long lPOrdLineId, long lInvId,
long lQty);
```

Internal BASIC syntax

```
Function AmAddPOLineToInv(lPOrdLineId As Long, lInvId As Long, lQty As
Long) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	X
Wizard script	
FINISH.DO script of a wizard	X
AssetCenter Web APIs	X

Description

This function adds a given quantity of item(s) on an order line to a supplier invoice.

Input parameters

- *lPOrdLineId*: This parameter contains the identifier of the order line to be added to the supplier invoice.
- *lInvId*: This parameter contains the identifier of the supplier invoice to which the items of the order line are added.
- *lQty*: This parameter contains the quantity of items present on the order line to be added to the supplier invoice.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmAddReqLinesToEstim()

API syntax

```
long AmAddReqLinesToEstim(long hApiCnxBase, long lReqId, long lEstimId,
long bMergeLines);
```

Internal BASIC syntax

```
Function AmAddReqLinesToEstim(lReqId As Long, lEstimId As Long,
bMergeLines As Long) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	X
Wizard script	
FINISH.DO script of a wizard	X
AssetCenter Web APIs	X

Description

This function adds all request lines of a request to an existing estimate.

Input parameters

- `lReqId`: This parameter contains the identifier of the request to be added to the estimate.
- `lEstimId`: This parameter contains the identifier of the estimate to which all the request lines of the request are added.
- `bMergeLines`: This parameter allows you to specify if identical request lines are to be combined (`bMergeLines`=0) to given one single line. The quantities given for the lines to be combined are added together and a single line is created.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmAddReqLinesToPO()

API syntax

```
long AmAddReqLinesToPO(long hApiCnxBase, long lReqId, long lPOrdId,
long bMergeLines);
```

Internal BASIC syntax

```
Function AmAddReqLinesToPO(lReqId As Long, lPOrdId As Long, bMergeLines
As Long) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	X
Wizard script	
FINISH.DO script of a wizard	X
AssetCenter Web APIs	X

Description

This function adds all the request lines of a request to an existing purchase order. The supplier specified in the request must be identical to that in the purchase order concerned.

Input parameters

- *lReqId*: This parameter contains the identifier of the request to be added to the purchase order.
- *lPOrdId*: This parameter contains the identifier of the purchase order to which the request lines are to be added.
- *bMergeLines*: This parameter allows you to specify if identical request lines are to be combined (*bMergeLines*=0) to given one single line. The quantities given for the lines to be combined are added together and a single line is created.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmAddReqLineToEstim()

API syntax

```
long AmAddReqLineToEstim(long hApiCnxBase, long lReqLineId, long
lEstimId, long bMergeLines);
```

Internal BASIC syntax

```
Function AmAddReqLineToEstim(lReqLineId As Long, lEstimId As Long,
bMergeLines As Long) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X

	Available
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	X
Wizard script	
FINISH.DO script of a wizard	X
AssetCenter Web APIs	X

Description

This function adds a request line to an existing estimate.

Input parameters

- `lReqLineId`: This parameter contains the identifier of the request line to be added to the estimate.
- `lEstimId`: This parameter contains the identifier of the estimate to which the request line is added.
- `bMergeLines`: This parameter allows you to specify if identical request lines are to be combined (`bMergeLines`=0) to given one single line. The quantities given for the lines to be combined are added together and a single line is created.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmAddReqLineToPO()

API syntax

```
long AmAddReqLineToPO(long hApiCnxBase, long lReqLineId, long lPOrdId,
long bMergeLines);
```

Internal BASIC syntax

```
Function AmAddReqLineToPO(lReqLineId As Long, lPOrdId As Long,
bMergeLines As Long) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	X
Wizard script	
FINISH.DO script of a wizard	X
AssetCenter Web APIs	X

Description

This function adds a request line to an existing purchase order.

Input parameters

- *lReqLineId*: This parameter contains the identifier of the request line to be added to the purchase order.
- *lPOrdId*: This parameter contains the identifier of the purchase order to which the request line is to be added.
- *bMergeLines*: This parameter allows you to specify if identical request lines are to be combined (*bMergeLines*=0) to given one single line. The quantities given for the lines to be combined are added together and a single line is created.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmApproveRequest()

API syntax

```
long AmApproveRequest(long hApiCnxBase, long lReqId, char *strComment,
long bCheckCompo, long bCheckCost, double dMaxCost);
```

Internal BASIC syntax

```
Function AmApproveRequest(lReqId As Long, strComment As String,
bCheckCompo As Long, bCheckCost As Long, dMaxCost As Double) As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	

	Available
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	
Wizard script	
FINISH.DO script of a wizard	
AssetCenter Web APIs	

Description

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmBusinessSecondsInDay()

API syntax

```
long AmBusinessSecondsInDay(char *strCalendarSqlName, long tmDate);
```

Internal BASIC syntax

```
Function AmBusinessSecondsInDay(strCalendarSqlName As String, tmDate As
Date) As Date
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Calculates the number of business seconds in a day according to a calendar.

Input parameters

- `strCalendarSqlName`: SQL name of the calendar used for the calculation.
- `tmDate`: Date for which the calculation is performed.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmCalcConsolidatedFeature()

API syntax

```
long AmCalcConsolidatedFeature(long hApiCnxBase, long lCalcFeatId, char
*strSQLTableName);
```

Internal BASIC syntax

```
Function AmCalcConsolidatedFeature(lCalcFeatId As Long, strSQLTableName
As String) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Calculates the value of consolidated feature on a table identified by its SQL name.

Input parameters

- *lCalcFeatId*: Identifier of the consolidated feature.
- *strSQLTableName*: SQL name of the table for which the consolidated feature is calculated. The feature must be defined for this table.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmCalcDepr()

API syntax

```
double AmCalcDepr(long iType, long lDuration, double dCoeff, double
dPrice, long tmStart, long tmDate);
```

Internal BASIC syntax

```
Function AmCalcDepr(iType As Long, lDuration As Long, dCoeff As Double,
dPrice As Double, tmStart As Date, tmDate As Date) As Double
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X

	Available
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function allows you calculate the depreciation amount for an asset on a given date. It returns the depreciation value on this date.

Input parameters

- `iType`: This parameter identifies the nature of the depreciation. The following values are possible:
  - 0: No depreciation
  - 1: Straight line
  - 2: Declining balance
- `lDuration`: This parameter contains the period over which the asset is depreciated. This period is expressed in seconds.
- `dCoeff`: This parameter contains the coefficient applied in the declining balance method. It is not interpreted in the case of the straight line depreciation method but must have a value.
- `dPrice`: This parameter contains the initial value of the asset concerned by the depreciation calculation.
- `tmStart`: This parameter contains the date from which the asset is depreciated.
- `tmDate`: This parameter contains the date on which the depreciation and residual value of the asset are calculated.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.

- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmCleanup()

Internal BASIC syntax

```
Field of application
```

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	
Wizard script	
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function must be called at the end of scripts using the database modification functions. It frees all used resources.

AmClearLastError()

API syntax

```
long AmClearLastError(long hApiCnxBase);
```

Internal BASIC syntax

```
Function AmClearLastError() As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function clears the information concerning the last error message occurred in the current connection.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmCloseAllChildren()

API syntax

```
long AmCloseAllChildren(long hApiCnxBase);
```

Internal BASIC syntax

```
Function AmCloseAllChildren() As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function destroys all the objects created during the current connection.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmCloseConnection()

API syntax

```
long AmCloseConnection(long hApiCnxBase);
```

Internal BASIC syntax

```
Function AmCloseConnection() As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	X
Wizard script	
FINISH.DO script of a wizard	X

	Available
AssetCenter Web APIs	X

Description

Ends the AssetCenter session for a given connection. All objects (queries, records, tables, fields, etc.) created within this connection are automatically destroyed. Their handles become invalid.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmCommit()

API syntax

```
long AmCommit(long hApiCnxBase);
```

Internal BASIC syntax

```
Function AmCommit() As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	X

	Available
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	X
Wizard script	
FINISH.DO script of a wizard	X
AssetCenter Web APIs	X

Description

This function commits all the modifications made to the database associated with the connection.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmComputeAllLicAndInstallCounts()

API syntax

```
long AmComputeAllLicAndInstallCounts(long hApiCnxBase);
```

Internal BASIC syntax

```
Function AmComputeAllLicAndInstallCounts() As Long
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	X
Wizard script	
FINISH.DO script of a wizard	X
AssetCenter Web APIs	X

Description

This function performs the count of software licenses and installations for all records.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmComputeLicAndInstallCounts()

API syntax

```
long AmComputeLicAndInstallCounts(long hApiCnxBase, long lSLCountId);
```

Internal BASIC syntax

```
Function AmComputeLicAndInstallCounts(lSLCountId As Long) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	X
Wizard script	
FINISH.DO script of a wizard	X
AssetCenter Web APIs	X

Description

This function performs the count of software licenses and installations for a record.

Input parameters

- `lSLCountId`: This parameter contains the identifier of the software license counter.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmConvertCurrency()

API syntax

```
double AmConvertCurrency(long hApiCnxBase, long tmDate, char
*strSrcName, char *strDstName, double dVal);
```

Internal BASIC syntax

```
Function AmConvertCurrency(tmDate As Date, strSrcName As String,
strDstName As String, dVal As Double) As Double
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function performs a conversion between two currencies at a given date, with a given precision.

Input parameters

- *tmDate*: This parameter contains the conversion date. It allows you to know the conversion rate in effect on this date.
- *strSrcName*: This parameter contains the source currency for the conversion, i.e. the currency you want to convert.
- *strDstName*: This parameter contains the target currency for the conversion, i.e. the currency in which you want to express the source currency.
- *dVal*: This parameter contains the amount (expressed in the monetary unit of the source currency) to be converted.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Notes

The currency parameters (*strSrcName* and *strDstName*) for this function must be defined in AssetCenter. Furthermore, a valid exchange rate must exist for the date when you want to perform the conversion (*tmDate* parameter).

Example

The following example converts 5,000 FRF into dollars, on the date of November 02, 1998.

```
AmConvertCurrency("1998/11/02 00:00:00", "FRF", "$", 5000)
```

AmConvertDateBasicToUnix()

API syntax

```
long AmConvertDateBasicToUnix(long hApiCnxBase, long tmTime);
```

Internal BASIC syntax

```
Function AmConvertDateBasicToUnix(tmTime As Date) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function converts a Basic format date ("Date" type) to a Unix format date ("Long" type).

Input parameters

- *tmTime*: This parameter contains the date to be converted.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmConvertDateIntlToUnix()

API syntax

```
long AmConvertDateIntlToUnix(long hApiCnxBase, char *strDate);
```

Internal BASIC syntax

```
Function AmConvertDateIntlToUnix(strDate As String) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X

	Available
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function converts an international format date ("Date" type) to a Unix format date ("Long" type).

Input parameters

- `strDate`: This parameter contains the date to be converted.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmConvertDateStringToUnix()

API syntax

```
long AmConvertDateStringToUnix(long hApiCnxBase, char *strDate);
```

Internal BASIC syntax

```
Function AmConvertDateStringToUnix(strDate As String) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Converts a date to a string (as displayed in the Windows Control Panel) to a Unix "Long".

Input parameters

- `strDate`: Date as string to convert.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmConvertDateUnixToBasic()

API syntax

```
long AmConvertDateUnixToBasic(long hApiCnxBase, long lTime);
```

Internal BASIC syntax

```
Function AmConvertDateUnixToBasic(lTime As Long) As Date
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function converts a Unix format date ("Long" type) to a Basic format date ("Date" type)

Input parameters

- *lTime*: This parameter contains the date to be converted.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmConvertDateUnixToIntl()

API syntax

```
long AmConvertDateUnixToIntl(long hApiCnxBase, long lUnixDate, char
*pstrDate, long lDate);
```

Internal BASIC syntax

```
Function AmConvertDateUnixToIntl(lUnixDate As Long) As String
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X

	Available
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function converts a Unix format date ("Long" type) to an international format date ("Long" type).

Input parameters

- `lUnixDate`: This parameter contains the date to be converted.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmConvertDateUnixToString()

API syntax

```
long AmConvertDateUnixToString(long hApiCnxBase, long lUnixDate, char
*pstrDate, long lDate);
```

Internal BASIC syntax

```
Function AmConvertDateUnixToString(lUnixDate As Long) As String
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Converts a "Long" Unix format date to a string format date (as displayed in the Windows Control Panel).

Input parameters

- `lUnixDate`: "Long" Unix format date to convert.
-

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmConvertDoubleToString()

API syntax

```
long AmConvertDoubleToString(double dSrc, char *pstrDst, long lDst);
```

Internal BASIC syntax

```
Function AmConvertDoubleToString(dSrc As Double) As String
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function converts a double precision number to a string.

Input parameters

- *dSrc*: This parameter contains the double-precision number to be converted.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmConvertMonetaryToString()

API syntax

```
long AmConvertMonetaryToString(double dSrc, char *pstrDst, long lDst);
```

Internal BASIC syntax

```
Function AmConvertMonetaryToString(dSrc As Double) As String
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X

	Available
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function converts a monetary value to a character string.

Input parameters

- `dSrc`: This parameter contains the monetary value you want to convert.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmConvertStringToDouble()

API syntax

```
double AmConvertStringToDouble(char *strSrc);
```

Internal BASIC syntax

```
Function AmConvertStringToDouble(strSrc As String) As Double
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function converts a character string to a double precision number.

Input parameters

- `strSrc`: This parameter contains the character string to be converted.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmConvertStringToMonetary()

API syntax

```
double AmConvertStringToMonetary(char *strSrc);
```

Internal BASIC syntax

```
Function AmConvertStringToMonetary(strSrc As String) As Double
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function converts a character string to a monetary value.

Input parameters

- *strSrc*: This parameter contains the character string to be converted.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmCounter()

API syntax

```
long AmCounter(char *return, long lreturn, char *strCounterName, long
iWidth);
```

Internal BASIC syntax

```
Function AmCounter(strCounterName As String, iWidth As Long) As String
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	

	Available
"Script" type action	
Wizard script	
FINISH.DO script of a wizard	
AssetCenter Web APIs	

Description

This function returns the value of the `strCounterName` counter, incremented by 1, and expressed in `iWidth` digits. Leading zeros are added if `iWidth` is greater than the value of the counter.

Input parameters

- `strCounterName`: Name of the counter as it is defined in AssetCenter (access via the Tools/ Administration/ Counters menu item).
- `iWidth`: The value of this parameter forces the output format of the function to be expressed in n digits.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Example

The following example returns the value of the "Delivery" counter expressed in 5 digits:

```
AmCounter("Delivery", 5)
```

For example, if the "Delivery" counter equals "18", the function returns:

00019

AmCreateDelivFromPO()

API syntax

```
long AmCreateDelivFromPO(long hApiCnxBase, long lPOrdId);
```

Internal BASIC syntax

```
Function AmCreateDelivFromPO(lPOrdId As Long) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	X
Wizard script	
FINISH.DO script of a wizard	X
AssetCenter Web APIs	X

Description

This function receives a purchase order and returns the identifier of the receiving slip created.

Input parameters

- *lPOrdId*: This parameter contains the identifier of the purchase order to be received.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmCreateEstimFromReq()

API syntax

```
long AmCreateEstimFromReq(long hApiCnxBase, long lReqId, long lSuppId);
```

Internal BASIC syntax

```
Function AmCreateEstimFromReq(lReqId As Long, lSuppId As Long) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	X

	Available
Wizard script	
FINISH.DO script of a wizard	X
AssetCenter Web APIs	X

Description

This function creates an estimate from a purchase request and returns the identifier of the estimate created.

Input parameters

- *`lReqId`*: This parameter contains the identifier of the purchase request used to create the estimate.
- *`lSuppId`*: This parameter contains the identifier of the supplier of the estimate that will be created by the function.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmCreateEstimsFromAllReqLines()

API syntax

```
long AmCreateEstimsFromAllReqLines(long hApiCnxBase, long lReqId, long
bMergeLines, long lDefSuppId);
```

Internal BASIC syntax

```
Function AmCreateEstimsFromAllReqLines(lReqId As Long, bMergeLines As
Long, lDefSuppId As Long) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	X
Wizard script	
FINISH.DO script of a wizard	X
AssetCenter Web APIs	X

Description

This function creates an estimate from a request and returns the identifier of the estimate created.

Input parameters

- `lReqId`: This parameter contains the identifier of the request at the origin of the estimate.
- `bMergeLines`: This parameter allows you to specify if identical request lines are to be combined (`bMergeLines`=0) to given one single line. The quantities given for the lines to be combined are added together and a single line is created.
- `lDefSuppId`: This parameter contains the identifier of the default supplier for the estimate.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmCreateInvFromPO()

API syntax

```
long AmCreateInvFromPO(long hApiCnxBase, long lPOrdId);
```

Internal BASIC syntax

```
Function AmCreateInvFromPO(lPOrdId As Long) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	X
Wizard script	
FINISH.DO script of a wizard	X
AssetCenter Web APIs	X

Description

This function creates a supplier invoice from a purchase order and returns the identifier of the supplier invoice created.

Input parameters

- *lPOrdId*: This parameter contains the identifier of the purchase order at the origin of the invoice.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmCreateLink()

API syntax

```
long AmCreateLink(long hApiRecord, char *strLinkName, long
hApiRecDest);
```

Internal BASIC syntax

```
Function AmCreateLink(hApiRecord As Long, strLinkName As String,
hApiRecDest As Long) As Long
```

Field of application

Version: 3.00

	Available

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	X
Wizard script	
FINISH.DO script of a wizard	X
AssetCenter Web APIs	X

Description

This function modifies a link of a record and makes it point to a new record (`hApiRecDest`) in the target table.

Input parameters

- `hApiRecord`: This parameter contains the handle of the record containing the link to be modified.
- `strLinkName`: This parameter contains the SQL name of the link to be modified.
- `hApiRecDest`: This parameter contains a handle of the target record of the link.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmCreatePOFromEstim()

API syntax

```
long AmCreatePOFromEstim(long hApiCnxBase, long lEstimId);
```

Internal BASIC syntax

```
Function AmCreatePOFromEstim(lEstimId As Long) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	X
Wizard script	
FINISH.DO script of a wizard	X
AssetCenter Web APIs	X

Description

This function creates a purchase order from an estimate and returns the identifier of the purchase order created.

Input parameters

- *lEstimId*: This parameter contains the identifier of the estimate used to create the order.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmCreatePOFromReq()

API syntax

```
long AmCreatePOFromReq(long hApiCnxBase, long lReqId, long lSuppId);
```

Internal BASIC syntax

```
Function AmCreatePOFromReq(lReqId As Long, lSuppId As Long) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	X

	Available
Wizard script	
FINISH.DO script of a wizard	X
AssetCenter Web APIs	X

Description

This function creates a purchase order from a purchase request and returns the identifier of the PO created.

Input parameters

- `lReqId`: This parameter contains the identifier of the purchase request used to create the purchase order.
- `lSuppId`: This parameter contains the identifier of the supplier of the purchase order that will be created by the function.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmCreatePOsFromAllReqLines()

API syntax

```
long AmCreatePOsFromAllReqLines(long hApiCnxBase, long lReqId, long
bMergeLines, long lDefSuppId);
```

Internal BASIC syntax

```
Function AmCreatePOsFromAllReqLines(lReqId As Long, bMergeLines As
Long, lDefSuppId As Long) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	X
Wizard script	
FINISH.DO script of a wizard	X
AssetCenter Web APIs	X

Description

This function creates all the purchase orders from the request lines of a request.

Input parameters

- `lReqId`: This parameter contains the identifier of the request from which the purchase orders are to be created.
- `bMergeLines`: This parameter allows you to specify if identical request lines are to be combined (`bMergeLines`=0) to given one single line. The quantities given for the lines to be combined are added together and a single line is created.
- `lDefSuppId`: This parameter contains the identifier of the default supplier for the requested items. This parameter is optional and is set to "0" by default.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmCreateRecord()

API syntax

```
long AmCreateRecord(long hApiCnxBase, char *strTable);
```

Internal BASIC syntax

```
Function AmCreateRecord(strTable As String) As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	X
Wizard script	
FINISH.DO script of a wizard	X
AssetCenter Web APIs	X

Description

This function creates an empty record in a table taking the default values into account. This new record does not exist in the database until it has been inserted.

Input parameters

- `strTable`: This parameter contains the SQL name of the table in which you want to create the record.

AmCurrentDate()

API syntax

```
long AmCurrentDate();
```

Internal BASIC syntax

```
Function AmCurrentDate() As Date
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	

	Available
AssetCenter Web APIs	X

Description

This function returns the current date on the client workstation.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmCurrentLanguage()

API syntax

```
long AmCurrentLanguage(char *pstrLanguage, long lLanguage);
```

Internal BASIC syntax

```
Function AmCurrentLanguage() As String
```

Field of application

Version: 2.52

	Available
Built-in function	

	Available
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns the language version of AssetCenter ("US" for English, "FR" for French, etc.).

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmCurrentServerDate()

API syntax

```
long AmCurrentServerDate(long hApiCnxBase);
```

Internal BASIC syntax

```
Function AmCurrentServerDate() As Date
```

Field of application

Version: 3.5

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns the current date on the server.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmDateAdd()

API syntax

```
long AmDateAdd(long tmStart, long tsDuration);
```

Internal BASIC syntax

```
Function AmDateAdd(tmStart As Date, tsDuration As Long) As Date
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function calculates a new date according to a start date to which a real duration is added.

Input parameters

- *tmStart*: This parameter contains the date to which the duration is added.
- *tsDuration*: This parameter contains the duration to be added to the date *tmStart*.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmDateAddLogical()

API syntax

```
long AmDateAddLogical(long tmStart, long tsDuration);
```

Internal BASIC syntax

```
Function AmDateAddLogical(tmStart As Date, tsDuration As Long) As Date
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X

	Available
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function calculates a new date according to a start date to which a logical duration is added (1 month contains 30 days).

Input parameters

- `tmStart`: This parameter contains the date to which the duration is added.
- `tsDuration`: This parameter contains the duration to be added to the date `tmStart`.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmDateDiff()

API syntax

```
long AmDateDiff(long tmEnd, long tmStart);
```

Internal BASIC syntax

```
Function AmDateDiff(tmEnd As Date, tmStart As Date) As Date
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function calculates in the seconds the duration (or timespan) between two dates.

Input parameters

- *tmEnd*: This parameter contains the end date of the period for which the calculation is carried out.
- *tmStart*: This parameter contains the start date of the period for which the calculation is carried out.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Example

The following example calculates the time elapsed between 01/01/98 and 01/01/99.

```
AmDateDiff("1998/01/01 00:00:00", "1999/01/01 00:00:00")
```

AmDbGetDate()

API syntax

```
long AmDbGetDate(long hApiCnxBase, char *strQuery);
```

Internal BASIC syntax

```
Function AmDbGetDate(strQuery As String) As Date
```

Field of application

Version: 3.5

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns the result, in date format, of the AQL query.

Input parameters

- `strQuery`: This parameter contains the full AQL query whose result you want to recover.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmDbGetDouble()

API syntax

```
double AmDbGetDouble(long hApiCnxBase, char *strQuery);
```

Internal BASIC syntax

```
Function AmDbGetDouble(strQuery As String) As Double
```

Field of application

Version: 3.5

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns the result (as a double-precision number), of the AQL query.

Input parameters

- *strQuery*: This parameter contains the full AQL query whose result you want to recover.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmDbGetList()

API syntax

```
long AmDbGetList(long hApiCnxBase, char *strQuery, char *pstrResult,
long lResult, char *strColSep, char *strLineSep, char *strIdSep);
```

Internal BASIC syntax

```
Function AmDbGetList(strQuery As String, strColSep As String,
strLineSep As String, strIdSep As String) As String
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X

	Available
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns, as a list, the result of an AQL query. The number of elements selected by the AQL query is limited to 99.

Input parameters

- `strQuery`: This parameter contains the AQL query you want to execute.
- `strColSep`: This parameter contains the character used as column separator in the result given by the function.
- `strLineSep`: This parameter contains the character used as line separator in the result given by the function.
- `strIdSep`: This parameter contains the character used as identifier separator in the result given by the function.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmDbGetListEx()

API syntax

```
long AmDbGetListEx(long hApiCnxBase, char *strQuery, char *pstrResult,
long lResult, char *strColSep, char *strLineSep, char *strIdSep);
```

Internal BASIC syntax

```
Function AmDbGetListEx(strQuery As String, strColSep As String,
strLineSep As String, strIdSep As String) As String
```

Field of application

Version: 3.5

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns, as a list, the result of an AQL query. Unlike the `AmDbGetList` function, this function is not limited in the number of elements selected by the AQL query.

Input parameters

- *strQuery*: This parameter contains the AQL query you want to execute.
- *strColSep*: This parameter contains the character used as column separator in the result given by the function.
- *strLineSep*: This parameter contains the character used as line separator in the result given by the function.
- *strIdSep*: This parameter contains the character used as identifier separator in the result given by the function.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmDbGetLong()

API syntax

```
long AmDbGetLong(long hApiCnxBase, char *strQuery);
```

Internal BASIC syntax

```
Function AmDbGetLong(strQuery As String) As Long
```

Field of application

Version: 3.00

	Available

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns the result of an AQL query.

Input parameters

- `strQuery`: This parameter contains the full AQL query whose result you want to recover.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Example

The following example returns the identifier of a product supplier:

```
AmDbGetLong("SELECT lSuppId FROM amProdSupp WHERE
lProdId="+Str([ProdId])+")
```

AmDbGetPk()

API syntax

```
long AmDbGetPk(long hApiCnxBase, char *strTableName, char *strWhere);
```

Internal BASIC syntax

```
Function AmDbGetPk(strTableName As String, strWhere As String) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns the primary key of a table according to the WHERE clause in an AQL query.

Input parameters

- `strTableName`: SQL name of the table whose primary key you want to recover.
- `strWhere`: WHERE clause in an AQL query.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmDbGetString()

API syntax

```
long AmDbGetString(long hApiCnxBase, char *strQuery, char *pstrResult,
long lResult, char *strColSep, char *strLineSep);
```

Internal BASIC syntax

```
Function AmDbGetString(strQuery As String, strColSep As String,
strLineSep As String) As String
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X

	Available
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns the result of an AQL query as a formatted string. The number of elements selected by the AQL query is limited to 99.

Input parameters

- `strQuery`: This parameter contains the AQL query you want to execute.
- `strColSep`: This parameter contains the character used as column separator in the final string.
- `strLineSep`: This parameter contains the character used as line separator in the final string.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmDbGetStringEx()

API syntax

```
long AmDbGetStringEx(long hApiCnxBase, char *strQuery, char
*pstrResult, long lResult, char *strColSep, char *strLineSep);
```

Internal BASIC syntax

```
Function AmDbGetStringEx(strQuery As String, strColSep As String,
strLineSep As String) As String
```

Field of application

Version: 3.5

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns, as a character string , the result of an AQL query. The difference with the `AmDbGetString` function is that this function is not limited in the number of elements selected by the AQL query.

Input parameters

- *strQuery*: This parameter contains the AQL query you want to execute.
- *strColSep*: This parameter contains the character used as column separator in the final string.
- *strLineSep*: This parameter contains the character used as line separator in the final string.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmDeadLine()

API syntax

```
long AmDeadLine(char *strCalendarSqlName, long tmStart, long
tsDuration);
```

Internal BASIC syntax

```
Function AmDeadLine(strCalendarSqlName As String, tmStart As Date,
tsDuration As Long) As Date
```

Field of application

Version: 3.00

	Available
Built-in function	

	Available
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function calculates a deadline according to a calendar, a start date and a number of working seconds elapsed.

Input parameters

- `strCalendarSqlName`: This parameter contains the SQL name of the calendar of working periods used as a basis for calculating the deadline.
- `tmStart`: This parameter contains the start date of the period.
- `tsDuration`: This parameter contains the number of working seconds since the start date of the period.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Example

The following example calculates the deadline according to the calendar whose SQL name is "Calendar_Paris", from a period start date set to 01/09/1998 at 8 a.m. and for a number of seconds equal to 450,000.

```
AmDeadLine("Calendar_Paris", "1998/09/01 08:00:00", 450000)
```

This example returns the deadline, i.e. 22/09/1998 at 6 p.m.

AmDecrementLogLevel()

API syntax

```
long AmDecrementLogLevel();
```

Internal BASIC syntax

```
Function AmDecrementLogLevel() As Long
```

Field of application

Version: 3.5

	Available
Built-in function	
AssetCenter APIs	
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	

Description

This function allows you to go up one level in the hierarchy of a history window.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmDefaultCurrency()

Internal BASIC syntax

```
Function AmDefaultCurrency() As String
```

Field of application

Version: 3.5

	Available
Built-in function	?
AssetCenter APIs	
Configuration script of a field or link	?
Definition script of a feature parameter	?
"Script" type action	?
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	?

Description

Returns the default currency used in AssetCenter.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmDeleteLink()

API syntax

```
long AmDeleteLink(long hApiRecord, char *strLinkName, long
hApiRecDest);
```

Internal BASIC syntax

```
Function AmDeleteLink(hApiRecord As Long, strLinkName As String,
hApiRecDest As Long) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X

	Available
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	X
Wizard script	
FINISH.DO script of a wizard	X
AssetCenter Web APIs	X

Description

This function deletes a links of a record.

Input parameters

- `hApiRecord`: This parameter contains the handle of the record containing the link to be deleted.
- `strLinkName`: This parameter contains the SQL name of the link to be deleted.
- `hApiRecDest`: This parameter contains a handle of the target record of the link.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmDeleteRecord()

API syntax

```
long AmDeleteRecord(long hApiRecord);
```

Internal BASIC syntax

```
Function AmDeleteRecord(hApiRecord As Long) As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	X
Wizard script	
FINISH.DO script of a wizard	X
AssetCenter Web APIs	X

Description

This function deletes a record in the database.

Input parameters

- `hApiRecord`: This parameter contains a handle of the record you want to delete.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmEndOfNthBusinessDay()

API syntax

```
long AmEndOfNthBusinessDay(char *strCalendarSqlName, long tmStart, long
lDayCount);
```

Internal BASIC syntax

```
Function AmEndOfNthBusinessDay(strCalendarSqlName As String, tmStart As
Date, lDayCount As Long) As Date
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Gives the last business hour of the nth day (identified by the integer `lDayCount`) from a given date according to a calendar.

Input parameters

- *strCalendarSqlName*: Name of the calendar used for the calculation.
- *tmStart*: Start date for the calculation.
- *lDayCount*: Number of full business days to add to *tmStart* for the calculation.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmEnumValList()

API syntax

```
long AmEnumValList(long hApiCnxBase, char *strTableSqlName, char
*strEnumName, char *pstrValList, long lValList, long bNoCase, char
*strLineSep);
```

Internal BASIC syntax

```
Function AmEnumValList(strTableSqlName As String, strEnumName As
String, bNoCase As Long, strLineSep As String) As String
```

Field of application

Version: 3.5

	Available
Built-in function	

	Available
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns a string containing all the values of a system itemized list. The different values are delimited by the separator indicated in the `strLineSep` parameter.

If an itemized list value contains the character used as the separator or a "\", the "\" prefix is used.

Input parameters

- `strTableSqlName`: This parameter contains the SQL name of the table containing the system itemized-list.
- `strEnumName`: This parameter contains the SQL name of the system-itemized list for which you want to recover the values.
- `bNoCase`: This parameter ????
- `strLineSep`: This parameter contains the character used to delimit the itemized-list values.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.

- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmExecTransition()

API syntax

```
long AmExecTransition(char *strTransName);
```

Internal BASIC syntax

```
Function AmExecTransition(strTransName As String) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	

Description

This function triggers a valid transition from the current page.

Input parameters

- `strTransName`: This parameter contains the name of the transition as defined in the wizard script.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmExecuteActionById()

API syntax

```
long AmExecuteActionById(long lActionId, char *strTableName, long
lRecordId);
```

Internal BASIC syntax

```
Function AmExecuteActionById(lActionId As Long, strTableName As String,
lRecordId As Long) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	
Definition script of a feature parameter	

	Available
"Script" type action	X
Wizard script	
FINISH.DO script of a wizard	X
AssetCenter Web APIs	X

Description

This function executes an action as identified by its identifier.

Input parameters

- `lActionId`: This parameter contains the identifier of the action to be executed.
- `strTableName`: In the case of a contextual action, this parameter contains the SQL name of the table on which the action is executed. If this parameter is omitted, in the case of a contextual action, the function will fail. For non-contextual actions, this parameter is not interpreted and therefore optional.
- `lRecordId`: This parameter contains the identifier of a possible record concerned by the action. For non-contextual actions, this parameter is not interpreted and therefore optional.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmExecuteActionByName()

API syntax

```
long AmExecuteActionByName(char *strSqlName, char *strTableName, long
lRecordId);
```

Internal BASIC syntax

```
Function AmExecuteActionByName(strSqlName As String, strTableName As
String, lRecordId As Long) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	X
Wizard script	
FINISH.DO script of a wizard	X
AssetCenter Web APIs	X

Description

This function executes an action as identified by its SQL name.

Input parameters

- `strSqlName`: This parameter contains the SQL name of the action to be executed.
- `strTableName`: In the case of a contextual action, this parameter contains the SQL name of the table on which the action is executed. If this parameter is omitted, in the case of a contextual action, the function will fail. For non-contextual actions, this parameter is not interpreted and therefore optional.
- `lRecordId`: This parameter contains the identifier of a possible record concerned by the action. For non-contextual actions, this parameter is not interpreted and therefore optional.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmGenSqlName()

API syntax

```
long AmGenSqlName(char *return, long lreturn, char *strText);
```

Internal BASIC syntax

```
Function AmGenSqlName(strText As String) As String
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X

	Available
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function generates a valid SQL name from a classic string. Spaces are replaced by underscores ("_"). This function is especially useful for defining the default value of a SQL name for a feature based on its name.

Input parameters

- *`strText`*: Character string used to generate the SQL name.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Example

The following example defines the default value of the SQL name of an object called "Label" in the AssetCenter database:

```
RetVal=AmSQLName([Label])
```

AmGetComputeString()

API syntax

```
long AmGetComputeString(long hApiCnxBase, char *strTableName, long
lRecordId, char *strTemplate, char *pstrComputeString, long
lComputeString);
```

Internal BASIC syntax

```
Function AmGetComputeString(strTableName As String, lRecordId As Long,
strTemplate As String) As String
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns the description string of a given record according to a template.

Input parameters

- `strTableName`: This parameter contains the SQL name of the table of the record for which you want to recover the description string.
- `lRecordId`: This parameter contains the identifier of the record within the table.
- `strTemplate`: This parameter contains, in the form of a character string, the template used for the description string.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmGetFeat()

Internal BASIC syntax

```
Function AmGetFeat(hApiTable As Long, lPos As Long) As Long
```

Field of application

Version: 3.5

	Available
Built-in function	?
AssetCenter APIs	
Configuration script of a field or link	?
Definition script of a feature parameter	?

	Available
"Script" type action	?
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	?

Description

This function creates a feature object based on its name and returns the handle of the feature object created.

Input parameters

- `hApiTable`: This parameter contains a handle of a table.
- `lPos`: This parameter contains the position of the feature in the table.

AmGetFeatCount()

Internal BASIC syntax

```
Function AmGetFeatCount(hApiTable As Long) As Long
```

Field of application

Version: 3.5

	Available
Built-in function	?
AssetCenter APIs	

	Available
Configuration script of a field or link	?
Definition script of a feature parameter	?
"Script" type action	?
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	?

Description

This function returns the number of features of the table specified in the `hApiTable` parameter.

Input parameters

- `hApiTable`: This parameter contains a handle of a table.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmGetField()

API syntax

```
long AmGetField(long hApiObject, long lPos);
```

Internal BASIC syntax

```
Function AmGetField(hApiObject As Long, lPos As Long) As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function creates a field object from the handle of a query, a record or a table and returns the handle of the field object created.

Input parameters

- `hApiObject`: This parameter contains a handle of a query, record, or table.
- `lPos`: This parameter contains the position of the field (its index) within the object.

AmGetFieldBlobLength()

API syntax

```
long AmGetFieldBlobLength(long hApiRecord, long lFieldId);
```

Internal BASIC syntax

```
Function AmGetFieldBlobLength(hApiRecord As Long, lFieldId As Long) As
Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Retrieves the length of a blob field. A blob is a database binary value, for example, an image.

Input parameters

- *hApiRecord*: Valid record handle.
- *lFieldId*: Identifier of the field concerned by the operation.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmGetFieldBlobValue()

API syntax

```
long AmGetFieldBlobValue(long hApiRecord);
```

Internal BASIC syntax

```
Function AmGetFieldBlobValue(hApiRecord As Long) As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X

	Available
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Retrieves the value of the blob field. This function must be called just after `AmGetFieldBlobLength`.

Input parameters

- *`hApiRecord`*: Valid record handle.

AmGetFieldCount()

API syntax

```
long AmGetFieldCount(long hApiObject);
```

Internal BASIC syntax

```
Function AmGetFieldCount(hApiObject As Long) As Long
```

Field of application

Version: 2.52

	Available
Built-in function	

	Available
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns the number of fields contained in the current object.

Input parameters

- `hApiObject`: This parameter contains a handle of a valid record, query or table.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmGetFieldDateValue()

API syntax

```
long AmGetFieldDateValue(long hApiObject, long lFieldPos);
```

Internal BASIC syntax

```
Function AmGetFieldDateValue(hApiObject As Long, lFieldPos As Long) As
Date
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns the value of a field contained in the current object. This value is returned in "Date" format.

Input parameters

- *hApiObject*: This parameter contains a handle of a query or record.
- *lFieldPos*: This parameter contains the number of the field inside the current object.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmGetFieldDescription()

API syntax

```
long AmGetFieldDescription(long hApiField, char *pstrBuffer, long
lBuffer);
```

Internal BASIC syntax

```
Function AmGetFieldDescription(hApiField As Long) As String
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X

	Available
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns, as a character string ("String" format), the long description of a field identified by a handle.

Input parameters

- `hApiField`: This parameter contains a valid handle of the field whose long description you want to know.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmGetFieldDoubleValue()

API syntax

```
double AmGetFieldDoubleValue(long hApiObject, long lFieldPos);
```

Internal BASIC syntax

```
Function AmGetFieldDoubleValue(hApiObject As Long, lFieldPos As Long)
As Double
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns the value of a field contained in the current object. This value is returned in "Double" format.

Input parameters

- `hApiObject`: This parameter contains a handle of a query or record.
- `lFieldPos`: This parameter contains the number of the field inside the current object.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmGetFieldFormat()

API syntax

```
long AmGetFieldFormat(long hApiField, char *pstrBuffer, long lBuffer);
```

Internal BASIC syntax

```
Function AmGetFieldFormat(hApiField As Long) As String
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function is useful when the value of the "UserType" of the field concerned is:

- System itemized list
- Custom itemized list
- Time span
- Table or field SQL name

The function returns the format of the "UserType", i.e.

UserType	Format returned by the function
System Itemized List	List of itemized list entries.
Custom Itemized List	Name of the itemized list associated with the field.
Time span	Display format.
Table or Field SQL Name	SQL name of the field that stores the SQL name of the table.

Input parameters

- *`hApiField`*: This parameter contains a valid handle of the field whose "UserType" you want to know.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmGetFieldFormatFromName()

API syntax

```
long AmGetFieldFormatFromName(long hApiCnxBase, char *strTableName,
char *strFieldName, char *pFieldFormat, long lpFieldFormat);
```

Internal BASIC syntax

```
Function AmGetFieldFormatFromName(strTableName As String, strFieldName
As String) As String
```

Field of application

Version: 3.5

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns the "UserType" format of a field, from its name.

Input parameters

- *strTableName*: This parameter contains the SQL name of the table containing the field concerned by the operation.
- *strFieldName*: This parameter contains the SQL name of the field.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmGetFieldFromName()

API syntax

```
long AmGetFieldFromName(long hApiObject, char *strName);
```

Internal BASIC syntax

```
Function AmGetFieldFromName(hApiObject As Long, strName As String) As
Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X

	Available
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function creates a field object based on its name and returns the handle of the field object created.

Input parameters

- `hApiObject`: This parameter contains a handle of a query, record, or table.
- `strName`: This parameter contains the field name.

AmGetFieldLabel()

API syntax

```
long AmGetFieldLabel(long hApiField, char *pstrBuffer, long lBuffer);
```

Internal BASIC syntax

```
Function AmGetFieldLabel(hApiField As Long) As String
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns, as a character string ("String" format), the label of a field identified by a handle.

Input parameters

- `hApiField`: This parameter contains a valid handle of the field whose label you want to know.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmGetFieldLabelFromName()

API syntax

```
long AmGetFieldLabelFromName(long hApiCnxBase, char *strTableName, char
*strFieldName, char *pFieldLabel, long lpFieldLabel);
```

Internal BASIC syntax

```
Function AmGetFieldLabelFromName(strTableName As String, strFieldName
As String) As String
```

Field of application

Version: 3.5

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns the label of a field from its SQL name.

Input parameters

- *strTableName*: This parameter contains the SQL name of the table containing the field concerned by the operation.
- *strFieldName*: This parameter contains the SQL name of the field.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmGetFieldLongValue()

API syntax

```
long AmGetFieldLongValue(long hApiObject, long lFieldPos);
```

Internal BASIC syntax

```
Function AmGetFieldLongValue(hApiObject As Long, lFieldPos As Long) As
Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X

	Available
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns the value of a field contained in the current object.

Input parameters

- `hApiObject`: This parameter contains a handle of a query or record.
- `lFieldPos`: This parameter contains the number of the field inside the current object.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Notes

If you use this function to recover the value of a field of date, time or date+time type, the long integer returned by the function represents the number of seconds since 01/01/1970 at 00:00:00.

AmGetFieldName()

API syntax

```
long AmGetFieldName(long hApiObject, long lFieldPos, char *pstrBuffer,
long lBuffer);
```

Internal BASIC syntax

```
Function AmGetFieldName(hApiObject As Long, lFieldPos As Long) As
String
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns the name of a field contained in the current object.

Input parameters

- *hApiObject*: This parameter contains a handle of a query, record, or table.
- *lFieldPos*: This parameter contains the number of the field within the current object. E.g., the value "0" indicates the first field.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmGetFieldRights()

API syntax

```
long AmGetFieldRights(long hApiObject, long lFieldPos, char
*pstrBuffer, long lBuffer);
```

Internal BASIC syntax

```
Function AmGetFieldRights(hApiObject As Long, lFieldPos As Long) As
String
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	X

	Available
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns the user rights for a field in the current object. These rights are returned as a character string containing three characters, which specify the read/insert/update rights:

- "r": indicates the right to read data.
- "i": indicates the right to insert data.
- "u": indicates the right to update data.

For example, for a read-only field, the function returns the value "r ".

Input parameters

- `hApiObject`: This parameter contains a handle of a query, record, or table.
- `lFieldPos`: This parameter contains the number of the field within the current object.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmGetFieldSize()

API syntax

```
long AmGetFieldSize(long hApiField);
```

Internal BASIC syntax

```
Function AmGetFieldSize(hApiField As Long) As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns the size of a field.

Input parameters

- *hApiField*: This parameter contains a handle of the field whose size you want to know.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmGetFieldSqlName()

API syntax

```
long AmGetFieldSqlName(long hApiField, char *pstrBuffer, long lBuffer);
```

Internal BASIC syntax

```
Function AmGetFieldSqlName(hApiField As Long) As String
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X

	Available
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns, as a character string ("String" format), the SQL name of a field identified by a handle.

Input parameters

- `hApiField`: This parameter contains a valid handle of the field whose SQL name you want to know.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmGetFieldStrValue()

API syntax

```
long AmGetFieldStrValue(long hApiObject, long lFieldPos, char
*pstrBuffer, long lBuffer);
```

Internal BASIC syntax

```
Function AmGetFieldStrValue(hApiObject As Long, lFieldPos As Long) As
String
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns the value of a field contained in the current object. This value is returned in string format.

Warning: When this function is used via the AssetCenter APIs, it expects two extra parameters `pszBuffer` and `lBuffer`, which define a string used as a buffer to store the recovered string and the size of this buffer respectively. The `pszBuffer` string must be formatted (filled with characters) and be of the size defined by `lBuffer`. The following portion of code is incorrect, the string used as a buffer is not sized:

```
Dim strBuffer as String
Dim lRec as Long
Dim lBuffer as Long
lBuffer=20
lRec=AmGetFieldStrValue(1, 0, strBuffer, lBuffer)
```

Here is the corrected portion of code:

```
Dim strBuffer as String
Dim lRec as Long
Dim lBuffer as Long
strBuffer=String(21, " ") ' The buffer is set to 21 characters
(" ")
lBuffer=20
lRec=AmGetFieldStrValue(1, 0, strBuffer, lBuffer)
```

When you format a buffer string using the "String" function, do not use "0" as a padding character. Size the buffer before calling the `AmGetFieldStrValue` function, particularly if this function is in a loop and always uses the same string as a buffer.

Input parameters

- *`hApiObject`*: This parameter contains a handle of a query or record.
- *`lFieldPos`*: This parameter contains the number of the field inside the current object.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmGetFieldType()

API syntax

```
long AmGetFieldType(long hApiField);
```

Internal BASIC syntax

```
Function AmGetFieldType(hApiField As Long) As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns the type of a field.

Input parameters

- `hApiField`: This parameter contains a handle of the field whose type you want to know.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Notes

The following table lists the values returned by the `AmGetFieldType` function for each type of field:

Values returned	Corresponding field type
0	Undefined
1	Byte
2	Short
3	Long
4	Float
5	Double
6	String
7	Time stamp
8	Bin
9	Blob
10	Date
11	Time
12	Memo

AmGetFieldUserType()

API syntax

```
long AmGetFieldUserType(long hApiField);
```

Internal BASIC syntax

```
Function AmGetFieldUserType(hApiField As Long) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns the "UserType" of a field identified by a handle, in the form of a long integer. The valid return values are summarized below:

Stored value	Description
0	Default
1	Number
2	Yes/ No
3	Money
4	Date
5	Date+Time
7	System itemized list
8	Custom itemized list
10	Percentage
11	Time span

Stored value	Description
12	Table or field SQL name

Input parameters

- *hApiField*: This parameter contains a valid handle of the field whose "UserType" you want to know.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmGetForeignKey()

API syntax

```
long AmGetForeignKey(long hApiField);
```

Internal BASIC syntax

```
Function AmGetForeignKey(hApiField As Long) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	

	Available
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Recovers the handle of the foreign key of a link, itself identified by its handle.

Input parameters

- `hApiField`: Handle of the link concerned by the operation.

AmGetIndex()

Internal BASIC syntax

```
Function AmGetIndex(hApiTable As Long, lPos As Long) As Long
```

Field of application

Version: 3.5

	Available
Built-in function	?

	Available
AssetCenter APIs	
Configuration script of a field or link	?
Definition script of a feature parameter	?
"Script" type action	?
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	?

Description

This function creates an index object from a handle of a query, record, or a table and returns the handle of the index object created.

Input parameters

- `hApiTable`: This parameter contains a handle of a table.
- `lPos`: This parameter contains the position of the index in the table.

AmGetIndexCount()

Internal BASIC syntax

```
Function AmGetIndexCount(hApiTable As Long) As Long
```

Field of application

Version: 3.5

	Available
Built-in function	?
AssetCenter APIs	
Configuration script of a field or link	?
Definition script of a feature parameter	?
"Script" type action	?
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	?

Description

This function returns the number of indexes contained in the table specified in the `hApiTable` parameter.

Input parameters

- `hApiTable`: This parameter contains a handle of a table.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmGetIndexField()

Internal BASIC syntax

```
Function AmGetIndexField(hApiIndex As Long, lPos As Long) As Long
```

Field of application

Version: 3.5

	Available
Built-in function	?
AssetCenter APIs	
Configuration script of a field or link	?
Definition script of a feature parameter	?
"Script" type action	?
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	?

Description

AmGetIndexFieldCount()

Internal BASIC syntax

```
Function AmGetIndexFieldCount(hApiIndex As Long) As Long
```

Field of application

Version: 3.5

	Available
Built-in function	?
AssetCenter APIs	
Configuration script of a field or link	?
Definition script of a feature parameter	?
"Script" type action	?
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	?

Description

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmGetIndexFlags()

Internal BASIC syntax

```
Function AmGetIndexFlags(hApiIndex As Long) As Long
```

Field of application

Version: 3.5

	Available
Built-in function	?
AssetCenter APIs	
Configuration script of a field or link	?
Definition script of a feature parameter	?
"Script" type action	?
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	?

Description

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmGetIndexName()

Internal BASIC syntax

```
Function AmGetIndexName(hApiIndex As Long) As String
```

Field of application

Version: 3.5

	Available
Built-in function	?
AssetCenter APIs	
Configuration script of a field or link	?
Definition script of a feature parameter	?
"Script" type action	?
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	?

Description

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmGetLink()

API syntax

```
long AmGetLink(long hApiTable, long lPos);
```

Internal BASIC syntax

```
Function AmGetLink(hApiTable As Long, lPos As Long) As Long
```

Field of application

Version: 3.02

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function creates a link object from the handle of a table and returns the handle of the link object created.

Input parameters

- `hApiTable`: This parameter contains a handle of a table.
- `lPos`: This parameter contains the position of the link (its index) inside the object.

AmGetLinkCardinality()

Internal BASIC syntax

```
Function AmGetLinkCardinality(hApiField As Long) As Long
```

Field of application

Version: 3.5

	Available
Built-in function	?
AssetCenter APIs	
Configuration script of a field or link	?
Definition script of a feature parameter	?
"Script" type action	?
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	?

Description

This function returns the cardinality of a link.

Input parameters

- `hApiField`: This parameter contains a handle of the link whose cardinality you want to know.

Output parameters

- 1: The cardinality of the link is 1-1.

- 2: The cardinality of the link is 1-n.

AmGetLinkCount()

API syntax

```
long AmGetLinkCount(long hApiTable);
```

Internal BASIC syntax

```
Function AmGetLinkCount(hApiTable As Long) As Long
```

Field of application

Version: 3.02

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns the number of links contained in the current table.

Input parameters

- *hApiTable*: This parameter contains a handle of a valid table.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmGetLinkDstField()

Internal BASIC syntax

```
Function AmGetLinkDstField(hApiField As Long) As Long
```

Field of application

Version: 3.5

	Available
Built-in function	?
AssetCenter APIs	
Configuration script of a field or link	?
Definition script of a feature parameter	?
"Script" type action	?
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	?

Description

This function returns the field to which the link defined by the `hApiField` parameter points.

Input parameters

- `hApiField`: This parameter contains a handle of the link concerned by the operation.

AmGetLinkFeatureValue()

API syntax

```
long AmGetLinkFeatureValue(long hApiObject, long lFieldPos, long
lRecordId);
```

Internal BASIC syntax

```
Function AmGetLinkFeatureValue(hApiObject As Long, lFieldPos As Long,
lRecordId As Long) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	

	Available
AssetCenter Web APIs	X

Description

Returns the value of a "Link" type feature.

Input parameters

- *`hApiObject`*: This parameter contains a handle of a query or record.
- *`lFieldPos`*: This parameter contains the position of the field inside the current object.
- lRecordId: This parameter contains the identifier of the record whose feature value you want to recover.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Example

```
Dim q as String
```

q = "Select fv_link, lEmplDeptId From amEmplDept Where lEmplDeptId = " & [lEmplDeptId]

Dim hq as Long

hq = amQueryCreate()

Dim lErr as Long

lErr = amQueryGet(hq, q)

Dim lId as Long

```
lId = amGetFieldLongValue(hq, 1)
print "str: " & amGetFieldStrValue(hq, 0)
print "int: " &
amGetFieldLongValue(hq,0)
print "lnk: " & amGetLinkFeatureValue(hq,0,lId)
```

AmGetLinkFromName()

API syntax

```
long AmGetLinkFromName(long hApiTable, char *strName);
```

Internal BASIC syntax

```
Function AmGetLinkFromName(hApiTable As Long, strName As String) As
Long
```

Field of application

Version: 3.02

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function creates a link object from a name and returns the handle of the object created.

Input parameters

- *hApiTable*: This parameter contains a handle of a table.
- *strName*: This parameter contains the SQL name of the link.

AmGetLinkType()

API syntax

```
long AmGetLinkType(long hApiField);
```

Internal BASIC syntax

```
Function AmGetLinkType(hApiField As Long) As Long
```

Field of application

Version: 3.02

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X

	Available
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns the type of a link.

Input parameters

- `hApiField`: This parameter contains a handle of the link whose type you want to know.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmGetMainField()

API syntax

```
long AmGetMainField(long hApiTable);
```

Internal BASIC syntax

```
Function AmGetMainField(hApiTable As Long) As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function creates a field object corresponding to the main field in a given table. It returns a handle of the field thus created.

Input parameters

- `hApiTable`: This parameter contains a handle of the table whose main field you want to know.

AmGetRecordFromMainId()

API syntax

```
long AmGetRecordFromMainId(long hApiCnxBase, char *strTable, long lId);
```

Internal BASIC syntax

```
Function AmGetRecordFromMainId(strTable As String, lId As Long) As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	X
Wizard script	
FINISH.DO script of a wizard	X
AssetCenter Web APIs	X

Description

This function returns the ID number of a record identified by a value of the primary key of the table containing this record.

Input parameters

- `strTable`: This parameter contains the SQL name of the table containing the record concerned by the operation.
- `lId`: This parameter contains the value of the primary key of the table for this records.

AmGetRecordHandle()

API syntax

```
long AmGetRecordHandle(long hApiQuery);
```

Internal BASIC syntax

```
Function AmGetRecordHandle(hApiQuery As Long) As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	X
Wizard script	
FINISH.DO script of a wizard	X
AssetCenter Web APIs	X

Description

This function returns the handle of a record that is the current result of a query identified by its handle. This record can be used to write in the database.

Input parameters

- `hApiQuery`: This parameter contains a valid handle of a query object.

AmGetRecordId()

API syntax

```
long AmGetRecordId(long hApiRecord);
```

Internal BASIC syntax

```
Function AmGetRecordId(hApiRecord As Long) As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	X
Wizard script	
FINISH.DO script of a wizard	X
AssetCenter Web APIs	X

Description

This function returns the ID number of a record identified by its handle.

Input parameters

- *hApiRecord*: This parameter contains a valid handle of the record whose ID number you want to know.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmGetReverseLink()

API syntax

```
long AmGetReverseLink(long hApiField);
```

Internal BASIC syntax

```
Function AmGetReverseLink(hApiField As Long) As Long
```

Field of application

Version: 3.02

	Available
Built-in function	
AssetCenter APIs	X

	Available
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns the handle of the reverse link specified by the handle contained in the `hApiField` parameter.

Input parameters

- `hApiField`: This parameter contains a handle of the link whose reverse link you want to know.

AmGetSelfFromMainId()

API syntax

```
long AmGetSelfFromMainId(long hApiCnxBase, char *strTableName, long
lId, char *pstrRecordDesc, long lRecordDesc);
```

Internal BASIC syntax

```
Function AmGetSelfFromMainId(strTableName As String, lId As Long) As
String
```

Field of application

Version: 3.5

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Returns the description string for a record in a given table.

Input parameters

- `strTableName`: This parameter contains the SQL name of the table containing record concerned by the operation.
- `lId`: This parameter contains the ID number concerned by the operation.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmGetSourceTable()

API syntax

```
long AmGetSourceTable(long hApiField);
```

Internal BASIC syntax

```
Function AmGetSourceTable(hApiField As Long) As Long
```

Field of application

Version: 3.02

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Returns the handle of the source table of the link indicted in the `hApiField` parameter.

Input parameters

- *hApiField*: This parameter contains a valid handle of the link whose source table you want to know.

Output parameters

In case of error, this function returns a non-valid handle (zero).

AmGetTable()

API syntax

```
long AmGetTable(long hApiCnxBase, long lPos);
```

Internal BASIC syntax

```
Function AmGetTable(lPos As Long) As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns the handle of a table identified by its position in the current connection.

Input parameters

- *lPos*: This parameter contains the position of the table in the current connection. Its values are comprised between "0" and `AmGetTableCount`.

Output parameters

In case of error, this function returns a non-valid handle (zero).

AmGetTableCount()

API syntax

```
long AmGetTableCount(long hApiCnxBase);
```

Internal BASIC syntax

```
Function AmGetTableCount() As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	X

	Available
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns the number of tables in the database concerned by the currency connection.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmGetTableDescription()

API syntax

```
long AmGetTableDescription(long hApiTable, char *pstrDesc, long lDesc);
```

Internal BASIC syntax

```
Function AmGetTableDescription(hApiTable As Long) As String
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns, as a character string ("String" format), the long description of a table identified by a handle.

Input parameters

- `hApiTable`: This parameter contains a valid handle of the table whose long description you want to know.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmGetTableFromName()

API syntax

```
long AmGetTableFromName(long hApiCnxBase, char *strName);
```

Internal BASIC syntax

```
Function AmGetTableFromName(strName As String) As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns the handle of a table identified by its SQL name in the current connection.

Input parameters

- `strName`: This parameter contains the SQL name of the table whose handle you want to recover.

Output parameters

In case of error, this function returns a non-valid handle (zero).

AmGetTableLabel()

API syntax

```
long AmGetTableLabel(long hApiTable, char *pstrLabel, long lLabel);
```

Internal BASIC syntax

```
Function AmGetTableLabel(hApiTable As Long) As String
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns, as a character string ("String" format), the label of a table identified by a handle.

Input parameters

- *hApiTable*: This parameter contains a valid handle of the table whose label you want to know.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmGetTableName()

API syntax

```
long AmGetTableName(long hApiTable, char *pstrBuffer, long lBuffer);
```

Internal BASIC syntax

```
Function AmGetTableName(hApiTable As Long) As String
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Returns the SQL name of a table as a character string.

Input parameters

- `hApiTable`: Valid handle of the table whose name you want to recover.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmGetTableRights()

API syntax

```
long AmGetTableRights(long hApiTable, char *pstrBuffer, long lBuffer);
```

Internal BASIC syntax

```
Function AmGetTableRights(hApiTable As Long) As String
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns, as a character string (

Input parameters

- *hApiTable*: This parameter contains a valid handle of the table for which you want to know the user rights.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmGetTableSqlName()

API syntax

```
long AmGetTableSqlName(long hApiTable, char *pstrBuffer, long lBuffer);
```

Internal BASIC syntax

```
Function AmGetTableSqlName(hApiTable As Long) As String
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X

	Available
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns, as a character string ("String" format), the SQL name of a table identified by a handle.

Input parameters

- *`hApiTable`*: This parameter contains a valid handle of the table whose SQL name you want to know.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmGetTargetTable()

API syntax

```
long AmGetTargetTable(long hApiField);
```

Internal BASIC syntax

```
Function AmGetTargetTable(hApiField As Long) As Long
```

Field of application

Version: ?

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Returns the SQL name of the target table of a link.

Input parameters

- `hApiField`: Handle of the link concerned by the operation.

Output parameters

In case of error, this function returns a non-valid handle (zero).

AmGetTypedLinkField()

API syntax

```
long AmGetTypedLinkField(long hApiField);
```

Internal BASIC syntax

```
Function AmGetTypedLinkField(hApiField As Long) As Long
```

Field of application

Version: 3.02

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Returns a handle of the field whose value is the SQL name of the target table of the typed link indicated in the `hApiField` parameter.

Input parameters

- `hApiField`: This parameter contains a valid handle of the typed link at the origin of the operation.

AmGetVersion()

API syntax

```
long AmGetVersion(char *pstrBuf, long lBuf);
```

Internal BASIC syntax

```
Function AmGetVersion() As String
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns the build number of AssetCenter in the form of a character string.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmHasAdminPrivilege()

API syntax

```
long AmHasAdminPrivilege(long hApiCnxBase);
```

Internal BASIC syntax

```
Function AmHasAdminPrivilege() As Long
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X

	Available
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns "TRUE" (=1) if the connected user has administration rights.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmIncrementLogLevel()

API syntax

```
long AmIncrementLogLevel(char *strMsg, long iType);
```

Internal BASIC syntax

```
Function AmIncrementLogLevel(strMsg As String, iType As Long) As Long
```

Field of application

Version: 3.5

	Available
Built-in function	
AssetCenter APIs	
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	

Description

This function displays the `strMsg` message in a history window and creates a node.

All the following messages appear in this node.

Input parameters

- `strMsg`: This parameter contains the text of the message to be displayed.
- `iType`: This parameter defines the icon associated with the message. The possible values are "1" for an error, "2" for a warning, and "4" for information.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmInsertRecord()

API syntax

```
long AmInsertRecord(long hApiRecord);
```

Internal BASIC syntax

```
Function AmInsertRecord(hApiRecord As Long) As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	X
Wizard script	
FINISH.DO script of a wizard	X
AssetCenter Web APIs	X

Description

This function inserts a record previously created in the database. Only those records created using the `AmCreateRecord` function can be inserted in the database. Records accessed using a query cannot be inserted.

Input parameters

- *hApiRecord*: This parameter contains a handle of the record you want to insert in the database.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmIsConnected()

API syntax

```
long AmIsConnected(long hApiCnxBase);
```

Internal BASIC syntax

```
Function AmIsConnected() As Long
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	X
Wizard script	
FINISH.DO script of a wizard	X

	Available
AssetCenter Web APIs	X

Description

This function tests whether the current connection is valid.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmIsFieldForeignKey()

API syntax

```
long AmIsFieldForeignKey(long hApiField);
```

Internal BASIC syntax

```
Function AmIsFieldForeignKey(hApiField As Long) As Long
```

Field of application

Version: 2.52

	Available
Built-in function	

	Available
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function allows you to determine whether a field is an foreign key in the database.

Input parameters

- `hApiField`: This parameter contains a handle of the field to be identified.

Output parameters

- 1: The field is a foreign key.
- 0: The field is not a foreign key.

AmIsFieldIndexed()

Internal BASIC syntax

```
Function AmIsFieldIndexed(hApiField As Long) As Long
```

Field of application

Version: 3.5

	Available
Built-in function	?
AssetCenter APIs	
Configuration script of a field or link	?
Definition script of a feature parameter	?
"Script" type action	?
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	?

Description

This function allows you to determine whether a field is indexed or not.

Input parameters

- `hApiField`: This parameter contains a handle of the field to be identified.

Output parameters

- 1: The field is indexed.
- 0: The field is not indexed.

AmIsFieldPrimaryKey()

API syntax

```
long AmIsFieldPrimaryKey(long hApiField);
```

Internal BASIC syntax

```
Function AmIsFieldPrimaryKey(hApiField As Long) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function allows you to determine whether a field is an primary key in the database.

Input parameters

- `hApiField`: This parameter contains a handle of the field to be identified.

Output parameters

- 1: The field is a primary key.
- 0: The field is not a primary key.

AmIsLink()

API syntax

```
long AmIsLink(long hApiField);
```

Internal BASIC syntax

```
Function AmIsLink(hApiField As Long) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	

	Available
AssetCenter Web APIs	X

Description

Determines whether the object identified by its handle is a link or a field.

Input parameters

- `hApiField`: Handle of the object concerned by the operation.

Output parameters

- 1: The object is a link.
- 0: The object is a field.

AmIsTypedLink()

API syntax

```
long AmIsTypedLink(long hApiField);
```

Internal BASIC syntax

```
Function AmIsTypedLink(hApiField As Long) As Long
```

Field of application

Version: 3.02

	Available
Built-in function	

	Available
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Determines if the object identified by its handle is a typed link or not.

Input parameters

- `hApiField`: Handle of the object concerned by the operation.

Output parameters

- 1: The object is a typed link.
- 0: The object is not a typed link.

AmLastError()

API syntax

```
long AmLastError(long hApiCnxBase);
```

Internal BASIC syntax

```
Function AmLastError() As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns the last error code generated by the last function executed in the context of the corresponding connection.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmLastErrorMsg()

API syntax

```
long AmLastErrorMsg(long hApiCnxBase, char *pstrBuffer, long lBuffer);
```

Internal BASIC syntax

```
Function AmLastErrorMsg() As String
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns the last error message occurred in the current connection.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmListToString()

API syntax

```
long AmListToString(char *return, long lreturn, char *strSource, char
*strColSep, char *strLineSep, char *strIdSep);
```

Internal BASIC syntax

```
Function AmListToString(strSource As String, strColSep As String,
strLineSep As String, strIdSep As String) As String
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	

	Available
AssetCenter Web APIs	X

Description

This function converts the result of a character string obtained via the `AmDbGetList` function to a character string that can be displayed in the same way as the `AmDbGetString` function.

Input parameters

- *`strSource`*: This parameter contains the character string to be converted.
- *`strColSep`*: This parameter contains the character used as column separator in the string to be converted.
- *`strLineSep`*: This parameter contains the character used as line separator in the string to be converted.
- *`strIdSep`*: This parameter contains the character used as identifier separator in the string to be converted.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmLog()

API syntax

```
long AmLog(char *strMessage, long iLogType);
```

Internal BASIC syntax

```
Function AmLog(strMessage As String, iLogType As Long) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	

Description

This function displays the `strMessage` message in a history window.

Input parameters

- `strMessage`: This parameter contains the text of the message to be displayed.
- `iLogType`: This parameter defines the icon associated with the message. The possible values are "1" for an error, "2" for a warning, and "4" for information.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

Example

```
AmLog("This is a message")
```

AmLoginId()

API syntax

```
long AmLoginId(long hApiCnxBase);
```

Internal BASIC syntax

```
Function AmLoginId() As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns the identifier of the connected user.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Example

The following example defines the identifier of the connected user as the default value for a database field:

```
RetVal=AmLoginId()
```

AmLoginName()

API syntax

```
long AmLoginName(long hApiCnxBase, char *return, long lreturn);
```

Internal BASIC syntax

```
Function AmLoginName() As String
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X

	Available
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns the login name of the connected user.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Example

The following example defines the login name of the connected user as the default value for a database field:

```
RetVal=AmLoginName()
```

AmMsgBox()

API syntax

```
long AmMsgBox(char *strMessage);
```

Internal BASIC syntax

```
Function AmMsgBox(strMessage As String) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	

Description

This function displays a dialog box containing a message.

Input parameters

- `strMessage`: This parameter contains the message displayed in the dialog box.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

Example

```
AmMsgBox("Move carried out")
```

AmOpenConnection()

API syntax

```
long AmOpenConnection(char *strDataSource, char *strUser, char
*strPwd);
```

Internal BASIC syntax

```
Function AmOpenConnection(strDataSource As String, strUser As String,
strPwd As String) As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	X
Wizard script	
FINISH.DO script of a wizard	X
AssetCenter Web APIs	X

Description

Creates a session from an AC database name. `strDataSource` should be a valid AssetCenter data source name (these AC database connections are listed in the login box of AssetCenter).

You can open several connections, in the same database or on different databases.

Input parameters

- `strDataSource`: Name of the data source.
- `strUser`: User name for the connection.
- `strPwd`: Password of the specified user.

AmPagePath()

API syntax

```
long AmPagePath(char *pstrPath, long lPath);
```

Internal BASIC syntax

```
Function AmPagePath() As String
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	
Wizard script	X
FINISH.DO script of a wizard	

	Available
AssetCenter Web APIs	

Description

This function returns a string containing the execution path of the wizard, i.e. the list of pages browsed. Backward jumps are ignored.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmProgress()

API syntax

```
long AmProgress(long iProgress);
```

Internal BASIC syntax

```
Function AmProgress(iProgress As Long) As Long
```

Field of application

Version: 3.00

	Available

	Available
Built-in function	
AssetCenter APIs	
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	

Description

This function displays a progress indicator representing a percentage.

Input parameters

- `iProgress`: This parameter contains the percentage of completion (between 0 and 100) used to define size of the progress indicator.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

Example

```
AmProgress(85)
```

This function displays a progress indicator representing 85% completion.

AmQueryCreate()

API syntax

```
long AmQueryCreate(long hApiCnxBase);
```

Internal BASIC syntax

```
Function AmQueryCreate() As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function creates a query object in the current connection. This object can then be used to send AQL statements to the database server.

AmQueryExec()

API syntax

```
long AmQueryExec(long hApiQuery, char *strQueryCommand);
```

Internal BASIC syntax

```
Function AmQueryExec(hApiQuery As Long, strQueryCommand As String) As
Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function executes an AQL query. It returns the first result of the query. The next result can be obtained via the `AmQueryNext` function.

When the query sent by this function returns a "Memo" type field the result is limited to 255 characters.

Input parameters

- `hApiQuery`: This parameter contains a valid handle of the object to which the AQL statements are sent.
- `strQueryCommand`: This parameter contains the body of the AQL query as a string.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmQueryGet()

API syntax

```
long AmQueryGet(long hApiQuery, char *strQueryCommand);
```

Internal BASIC syntax

```
Function AmQueryGet(hApiQuery As Long, strQueryCommand As String) As
Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X

	Available
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function executes an AQL query. It only returns the first result of the query.

Input parameters

- `hApiQuery`: This parameter contains a valid handle of the object to which the AQL statements are sent.
- `strQueryCommand`: This parameter contains the body of the AQL query as a string.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmQueryNext()

API syntax

```
long AmQueryNext(long hApiQuery);
```

Internal BASIC syntax

```
Function AmQueryNext(hApiQuery As Long) As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns the result of a query executed beforehand using the `AmQueryExec` function.

Input parameters

- *`hApiQuery`*: This parameter contains a valid handle of the object to which the AQL statements are sent.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmQueryStartTable()

API syntax

```
long AmQueryStartTable(long hApiQuery);
```

Internal BASIC syntax

```
Function AmQueryStartTable(hApiQuery As Long) As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns a handle of the table concerned by a query identified by its handle.

Input parameters

- *hApiQuery*: This parameter contains a valid handle of a query object.

Output parameters

In case of error, this function returns a non-valid handle (zero).

AmQueryStop()

API syntax

```
long AmQueryStop(long hApiQuery);
```

Internal BASIC syntax

```
Function AmQueryStop(hApiQuery As Long) As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function interrupts the execution of a query identified by its handle. This query must have been launched beforehand using the `AmQueryExec` function.

Input parameters

- *`hApiQuery`*: This parameter contains a valid handle of a query object.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmReceiveAllPOLines()

API syntax

```
long AmReceiveAllPOLines(long hApiCnxBase, long lPOrdId, long
lDelivId);
```

Internal BASIC syntax

```
Function AmReceiveAllPOLines(lPOrdId As Long, lDelivId As Long) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	

	Available
AssetCenter APIs	X
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	X
Wizard script	
FINISH.DO script of a wizard	X
AssetCenter Web APIs	X

Description

This function receives all the items on an order line (takes delivery in full).

Note: Warning: Delivery lines are created by an agent when the transaction is committed. You cannot access them beforehand.

Input parameters

- `lPOrdId`: This parameter contains the identifier of the order line containing the items to be received.
- `lDelivId`: This parameter contains the identifier of the receiving slip used to receive all the items present on the order line.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmReceivePOLine()

API syntax

```
long AmReceivePOLine(long hApiCnxBase, long lPOrdLineId, long lDelivId,
long lQty);
```

Internal BASIC syntax

```
Function AmReceivePOLine(lPOrdLineId As Long, lDelivId As Long, lQty As
Long) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	X
Wizard script	
FINISH.DO script of a wizard	X
AssetCenter Web APIs	X

Description

This function takes delivery of a certain quantity of items on an order line (takes delivery in part) and returns the identifier of the delivery line.

Note: Warning: The delivery lines are created by an agent as soon as the transaction is committed. You cannot access them until this is performed.

Input parameters

- *lPOrdLineId*: This parameter contains the identifier of the purchase order line containing the items to be received.
- *lDelivId*: This parameter contains the identifier of the receiving slip used to receive a certain quantity of items present on the order line.
- *lQty*: This parameter contains the quantity of items on the order line to be received in the receiving slip.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmRefreshAllCaches()

API syntax

```
long AmRefreshAllCaches(long hApiCnxBase);
```

Internal BASIC syntax

```
Function AmRefreshAllCaches() As Long
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function refreshes the caches used in AssetCenter.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmRefreshProperty()

API syntax

```
long AmRefreshProperty(char *strVarName);
```

Internal BASIC syntax

```
Function AmRefreshProperty(strVarName As String) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	

Description

Reevaluates the value of a property identified by the `strVarName` parameter. If this property uses a script, the script is executed again.

Otherwise the tree of dependencies is updated.

Input parameters

- `strVarName`: Name of the property (of the wizard) that you want to reevaluate.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmRefuseRequest()

API syntax

```
long AmRefuseRequest(long hApiCnxBase, long lReqId, char *strComment);
```

Internal BASIC syntax

```
Function AmRefuseRequest(lReqId As Long, strComment As String) As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	
Wizard script	
FINISH.DO script of a wizard	
AssetCenter Web APIs	

Description

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmReleaseHandle()

API syntax

```
long AmReleaseHandle(long hApiObject);
```

Internal BASIC syntax

```
Function AmReleaseHandle(hApiObject As Long) As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function frees the handle and sub-handles of an object.

Input parameters

- *hApiObject*: This parameter contains a handle of the object concerned.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmRgbColor()

API syntax

```
long AmRgbColor(char *strText);
```

Internal BASIC syntax

```
Function AmRgbColor(strText As String) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	

	Available
AssetCenter Web APIs	X

Description

This function gives the RGB value of the color corresponding to the `strText` parameter.

Input parameters

- `strText`: This parameter contains the name of the color:
  - White
  - LtGray
  - Gray
  - DkGray
  - Black
  - Red
  - Green
  - Blue
  - Yellow
  - Cyan
  - Magenta
  - DkYellow<
  - DkGreen
  - DkCyan
  - DkBlue
  - DkMagenta
  - DkRed

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.

- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmRollback()

API syntax

```
long AmRollback(long hApiCnxBase);
```

Internal BASIC syntax

```
Function AmRollback() As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	X
Wizard script	
FINISH.DO script of a wizard	X
AssetCenter Web APIs	X

Description

This function cancels all modifications made before the declaration of the start of the transaction (performed via the `AmStartTransaction` function).

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmSeCreateDefVal()

API syntax

```
long AmSeCreateDefVal(long hApiCnxBase, long iProductTypeNature);
```

Internal BASIC syntax

```
Function AmSeCreateDefVal(iProductTypeNature As Long) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	

	Available
AssetCenter Web APIs	X

Description

This function sets the default value of the "seCreate" field found in the following tables:

- amProdCompo
- amReqLine
- amPOrdLine
- amEstimLine

Input parameters

- `iProductTypeNature`: This parameter contains an integer representing the element in the associated system itemized list. The following table shows the system itemized list entries with their corresponding integers:

Itemized list element	Associated integer
Standard hardware	0
Computer	1
Software license	2
Word order	3
Contract	4
Standard configuration	5
Training	6
Other	7

Output parameters

- 0: Normal execution.

- Other than zero: Error code.

AmSetFieldDateValue()

API syntax

```
long AmSetFieldDateValue(long hApiRecord, char *strFieldName, long
tmValue);
```

Internal BASIC syntax

```
Function AmSetFieldDateValue(hApiRecord As Long, strFieldName As
String, tmValue As Date) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	X
Wizard script	
FINISH.DO script of a wizard	X
AssetCenter Web APIs	X

Description

This function modifies a field in a record. This function does not update the database.

Input parameters

- *hApiRecord*: This parameter contains the handle of the record containing the field to be modified.
- *strFieldName*: This parameter contains the SQL name of the field to be modified.
- *tmValue*: This parameter contains the new value of the field in "Date" format.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmSetFieldDoubleValue()

API syntax

```
long AmSetFieldDoubleValue(long hApiRecord, char *strFieldName, double
dValue);
```

Internal BASIC syntax

```
Function AmSetFieldDoubleValue(hApiRecord As Long, strFieldName As
String, dValue As Double) As Long
```

Field of application

Version: 3.00

	Available

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	X
Wizard script	
FINISH.DO script of a wizard	X
AssetCenter Web APIs	X

Description

This function modifies a field in a record. This function does not update the database.

Input parameters

- `hApiRecord`: This parameter contains the handle of the record containing the field to be modified.
- `strFieldName`: This parameter contains the SQL name of the field to be modified.
- `dValue`: This parameter contains the new value of the field in "Double" format.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmSetFieldLongValue()

API syntax

```
long AmSetFieldLongValue(long hApiRecord, char *strFieldName, long
lValue);
```

Internal BASIC syntax

```
Function AmSetFieldLongValue(hApiRecord As Long, strFieldName As
String, lValue As Long) As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	X
Wizard script	
FINISH.DO script of a wizard	X
AssetCenter Web APIs	X

Description

This function modifies a field in a record. This function does not update the database. To modify the value of a date, time or date+time date you must express the new value in terms of seconds elapsed since 01/01/1970 at 00:00:00.

Input parameters

- `hApiRecord`: This parameter contains the handle of the field that to be modified.
- `strFieldName`: This parameter contains the SQL name of the field to be modified.
- `lValue`: This parameter contains the new value of the field.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmSetFieldStrValue()

API syntax

```
long AmSetFieldStrValue(long hApiRecord, char *strFieldName, char
*strValue);
```

Internal BASIC syntax

```
Function AmSetFieldStrValue(hApiRecord As Long, strFieldName As String,
strValue As String) As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	
Definition script of a feature parameter	

	Available
"Script" type action	X
Wizard script	
FINISH.DO script of a wizard	X
AssetCenter Web APIs	X

Description

This function modifies a field in a record. This function does not update the database.

Input parameters

- `hApiRecord`: This parameter contains the handle of the record containing the field to be modified.
- `strFieldName`: This parameter contains the SQL name of the field to be modified.
- `strValue`: This parameter contains the new value of the field in "String" format.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmSetLinkFeatureValue()

API syntax

```
long AmSetLinkFeatureValue(long hApiRecord, char *strFeatSqlName, char
*strDstSelfValue, long lDstId);
```

Internal BASIC syntax

```
Function AmSetLinkFeatureValue(hApiRecord As Long, strFeatSqlName As
String, strDstSelfValue As String, lDstId As Long) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	X
Wizard script	
FINISH.DO script of a wizard	X
AssetCenter Web APIs	X

Description

This function sets the value of a link type feature for a given record.

Input parameters

- `hApiRecord`: This parameter contains the identifier of the record to which the link type feature is associated.
- `strFeatSqlName`: This parameter contains the SQL name of the link type feature whose value you want to set. This SQL name is always preceded by "fv_".
- `strDstSelfValue`: This parameter contains the value of the feature as it will be displayed for the record. It is the "Self" value of the record with identifier `lDstId`. If you pass an invalid or non-existent value, you take the risk of corrupting the integrity of the database.

- `lDstId`: This parameter contains the identifier of the record to which the link type feature points.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmSetProperty()

API syntax

```
long AmSetProperty(char *strVarName);
```

Internal BASIC syntax

```
Function AmSetProperty(strVarName As String, vValue As Variant) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	

Description

This function sets the value of a property identified by its name. It also updates the tree of dependencies of this property.

Input parameters

- *strVarName*: This parameter contains the name of the property whose value you want to set.
- *vValue*: This parameter contains the new value for the property.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmSetUseUserName()

API syntax

```
long AmSetUseUserName(long hApiCnxBase, long bUseUserName);
```

Internal BASIC syntax

```
Function AmSetUseUserName(bUseUserName As Long) As Long
```

Field of application

Version: 2.52

	Available
Built-in function	

	Available
AssetCenter APIs	
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	
Wizard script	
FINISH.DO script of a wizard	
AssetCenter Web APIs	

Description

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmStartTransaction()

API syntax

```
long AmStartTransaction(long hApiCnxBase);
```

Internal BASIC syntax

```
Function AmStartTransaction() As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	X
Wizard script	
FINISH.DO script of a wizard	X
AssetCenter Web APIs	X

Description

This function starts a new transaction with the database associated with the connection. The next "Commit" or "Rollback" statement will validate or cancel all the modifications made to the database.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmStartup()

Internal BASIC syntax

```
Field of application
```

Version: 2.52

	Available

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	
Wizard script	
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function must be applied before all other functions. It initializes calls to the AssetCenter library.

AmTableDesc()

API syntax

```
long AmTableDesc(long hApiCnxBase, char *return, long lreturn, char
*strSqlName);
```

Internal BASIC syntax

```
Function AmTableDesc(strSqlName As String) As String
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function generates a character string with the format "<Description of the table> (<SQL name of the table>)" from the SQL name of the table.

Input parameters

- `strSqlName`: SQL name of the table for which a description string is required. If this parameter contains an invalid SQL name, the function returns a question mark ("?").

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Example

The following example generates a description string for the table of assets (SQL name: amAsset):

```
AmTableDesc("amAsset")
```

The result is as follows:

```
Assets (amAsset)
```

AmTaxRate()

API syntax

```
double AmTaxRate(char *strTaxRateName, long lTaxLocId, long tmDate,
double dValue);
```

Internal BASIC syntax

```
Function AmTaxRate(strTaxRateName As String, lTaxLocId As Long, tmDate
As Date, dValue As Double) As Double
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function calculates a tax rate according to a tax type, tax jurisdiction and a date.

Input parameters

- *strTaxRateName*: This parameter contains the SQL name of the tax type used to calculate the tax type.
- *lTaxLocId*: This parameter contains the ID number of the tax jurisdiction concerned by the tax type.
- *tmDate*: This parameter contains the date for which you want to know the tax rate.
- *dValue*: ?

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

AmUpdateDetail()

API syntax

```
long AmUpdateDetail(char *strFieldName);
```

Internal BASIC syntax

```
Function AmUpdateDetail(strFieldName As String, varValue As Variant) As
Long
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	
Wizard script	
FINISH.DO script of a wizard	X
AssetCenter Web APIs	

Description

This function is used in the data-entry wizards. The context (table for which a record is updated or populated or updated using the wizard) is therefore clearly defined. The function updates or populates fields or links of the context according to a value.

Input parameters

- `strFieldName`: This parameter contains the SQL name of the feature to be updated.
- `varValue`: This parameter contains the new value of the field.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmUpdateLoginSlot()

API syntax

```
long AmUpdateLoginSlot(long hApiCnxBase);
```

Internal BASIC syntax

```
Function AmUpdateLoginSlot() As Long
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	X
Wizard script	
FINISH.DO script of a wizard	X
AssetCenter Web APIs	X

Description

This function forces the update of information concerning the connected user's login slot.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmUpdateRecord()

API syntax

```
long AmUpdateRecord(long hApiRecord);
```

Internal BASIC syntax

```
Function AmUpdateRecord(hApiRecord As Long) As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	X
Wizard script	
FINISH.DO script of a wizard	X
AssetCenter Web APIs	X

Description

This function allows you to update a record.

Input parameters

- *hApiRecord*: This parameter contains a handle of the record containing the field to be updated.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmwAdminLogout()

API syntax

```
long AmwAdminLogout();
```

Internal BASIC syntax

```
Function AmwAdminLogout() As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	

	Available
Wizard script	
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Terminates a session when the user is connected as the Web administrator.

Output parameters

- 0: Normal execution.
- Other than zero: Error code (accessible via the `AmwErrorMsg` function.

Example

```
iRc = AmwAdminLogout()
```

rem session is closed and nothing else can be done in script section

AmwApproveRequest()

API syntax

```
long AmwApproveRequest(long lReqId, char *strComment, long bCheckCompo,
long bCheckCost, double dMaxCost);
```

Internal BASIC syntax

```
Function AmwApproveRequest(lReqId As Long, strComment As String,
bCheckCompo As Long, bCheckCost As Long, dMaxCost As Double) As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	
Wizard script	
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Approves a purchase request identified by its ID number.

Input parameters

- `lReqId`: Identifier of the purchase request ("amReqApprLine" table).
- `strComment`: Approver's comments.
- `bCheckCompo`: If this parameter is set to "1", approval will be canceled if the composition of the purchase request changes.
- `bCheckCost`: If this parameter is set to "1", approval of the purchase request is canceled if the grand total exceeds the amount specified in the `dMaxCost` parameter.
- `dMaxCost`: Maximum cost of the request for approval. This parameter is only used if the `bCheckCost` parameter is set to "1".

Output parameters

- 0: Normal execution.

- Other than zero: Error code (accessible via the `AmwErrorMsg` function.

Example

```
If (AmwGetTpl("Action") = "Approve") Then
   AmwSetTpl "bApproved", "True"
   iRc = AmApproveRequest(lReqId, strComment, True, False, 0.0)
  Else
   AmwSetTpl "bApproved", "False"
   iRc = AmRefuseRequest(lReqId, strComment)
End If
```

The variable determines whether the user can approve or refuse the current request. The variable are supposed to be defined in another HTML template. The full source for the program is available in rqappr_s.htm, which is included in the standard templates, in the "websrv\htdocs\tpl\" subfolder of the AssetCenter installation folder.

AmwCloseSlave()

API syntax

```
long AmwCloseSlave(char *strId);
```

Internal BASIC syntax

```
Function AmwCloseSlave(strId As String) As Long
```

Field of application

Version: 2.52

	Available

	Available
Built-in function	
AssetCenter APIs	
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	
Wizard script	
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Forces disconnection of a user identified by their security key.

Input parameters

- `strId`: Identification string provided by the variable $$(id). This variable is created automatically for each template. Its value is that of the security key created on connection to the database.

Output parameters

- 0: Normal execution.
- Other than zero: Error code (accessible via the `AmwErrorMsg` function.

Example

```
SlaveId="3B24_4CA70BBD_2A50"
```

IRc = AmwCloseSlave(SlaveId)

AmwCurrentCnx()

API syntax

```
long AmwCurrentCnx();
```

Internal BASIC syntax

```
Function AmwCurrentCnx() As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	
Wizard script	
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Returns the identifier of the connection as a "Long". This identifier can also be used for the AssetCenter APIs.

Output parameters

- 0: Error code (accessible via the `AmwErrorMsg` function.
- Identifier of the current connection.

Example

```
lCnx = AmwCurrentCnx()
```

print "<H1> Current connection is: ";lCnx;"</H1>"

AmwCurrentUserId()

API syntax

```
long AmwCurrentUserId();
```

Internal BASIC syntax

```
Function AmwCurrentUserId() As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	
Wizard script	
FINISH.DO script of a wizard	

	Available
AssetCenter Web APIs	X

Description

Returns the identifier of the user (identifier of a record in the table of departments and employees) of the current session, as stored in the AssetCenter database.

User's with a valid login in the database are identified by a unique key. The value of this key is that of the "lEmplDeptId" field in the "amEmplDept" table.

Output parameters

Identifier of the user.

An error code is accessible via the `AmwErrorMsg` function.

Example

```
lUserId = AmwCurrentUserId()
```

print "<H1> Current user id:";lUserId;"</H1>"

AmwErrorMsg()

API syntax

```
long AmwErrorMsg(char *return, long lreturn);
```

Internal BASIC syntax

```
Function AmwErrorMsg() As String
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	
Wizard script	
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Returns the last error message.

Output parameters

Description of the error occurred during the last operation.

Example

```
iRc = AmwLogin(DbName,User,password)
```

If iRc <> 0 then print "Error:";AmwErrorMsg()

AmwExist()

API syntax

```
long AmwExist();
```

Internal BASIC syntax

```
Function AmwExist() As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	
Wizard script	
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Tests for a given variable in the templates.

Output parameters

- False: Variable not found.
- True: Variable found.

The error message can be accessed using the `AmwErrorMsg` function.

Example

```
Rem Set a template variable
```

AmwSetTpl "MyVar","Some value"

If AmwExist("Myvar") then print "MyVar=";AmwGetTpl("MyVar")

AmwFetchQuery()

API syntax

```
long AmwFetchQuery(char *strQuery, long iStart, long iNb);
```

Internal BASIC syntax

```
Function AmwFetchQuery(strQuery As String, iStart As Long, iNb As Long)
As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	
Wizard script	
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Queries the AssetCenter database and returns `iNb` items from the `iStart` position. The first position is "0".

The results are stored in the template data spaces using the automatically indexed variables. The fields concerned by the query are used in the name of these variables.

For example, if the query concerns the "Brand" field and 10 items from the database are returned by the query, 10 variables named "Brand0", ..., "Brand9" are created with their corresponding values.

Input parameters

- *strQuery*: Valid AQL query
- *iStart*: Position of the first element to recover (the first position is "0").
- *iNb*: Number of elements to recover.

Output parameters

- 0: Normal execution.
- Other than zero: Error code (accessible via the `AmwErrorMsg` function.

Example

```
<script Language="AmScript1.0">
iRc = AmwFetchQuery("select Name,Tel from amEmplDept",0,10
</script>

Name0 = $$(Name0) Tel0= $$(Tel0) <br>
Name1 = $$(Name1) Tel1= $$(Tel1) <br>
Etc...
```

In this example, the "Name" and "Tel." fields are recovered for the first 10 records in the table of departments and employees.

After execution, 10 variables "Name0", ..., "Name9" and the 10 variables "Tel.0", ..., "Tel.9" are created in the data space of the templates.

AmwGetEnv()

API syntax

```
long AmwGetEnv(char *return, long lreturn, char *strKeyName);
```

Internal BASIC syntax

```
Function AmwGetEnv(strKeyName As String) As String
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	
Wizard script	
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Returns as a string, the contents of the global environment variable.

Input parameters

- `strKeyName`: Name of the variable.

Output parameters

- Character string representing the contents of the global environment variable.
- An error code is accessible via the `AmwErrorMsg` function.

Example

```
MyVal = AmwGetEnv("aGlobalVarName")
```

Print "Global value for aGlobalVarName";MyVal

AmwGetTpl()

API syntax

```
long AmwGetTpl(char *return, long lreturn);
```

Internal BASIC syntax

```
Function AmwGetTpl() As String
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	
Wizard script	
FINISH.DO script of a wizard	

	Available
AssetCenter Web APIs	X

Description

Returns as a string the contents of the local template environment variable.

Output parameters

- Character string representing the contents of the local template environment variable.

An error code is accessible via the `AmwErrorMsg` function.

Example

```
MyVal = AmwGetTpl("aLocalVarName")
```

Print "Local value for aLocVarName"=;MyVal

AmwInclude()

API syntax

```
long AmwInclude();
```

Internal BASIC syntax

```
Function AmwInclude() As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	
Wizard script	
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Reads a file identified by its name and pastes it the current position in the template.

$$(AmwInclude("file.htm")) is an easy way to reuse common portions of HTML code in several templates.

Output parameters

- 0: Normal execution.
- Other than zero: Error code (accessible via the `AmwErrorMsg` function.

Example

```
<html>
```

<!"Use standard menubar -->

$$(AmwInclude("menubar.htm"))

AmWizChain()

API syntax

```
long AmWizChain(char *strWizSqlName);
```

Internal BASIC syntax

```
Function AmWizChain(strWizSqlName As String) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	

Description

This function executes a wizard B, inside a wizard A. When wizard B has finished executing, wizard A takes over again.

Input parameters

- `strWizSqlName`: SQL name of the wizard to be executed.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmwLoadFile()

API syntax

```
long AmwLoadFile();
```

Internal BASIC syntax

```
Function AmwLoadFile() As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	
Wizard script	
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Cancels evaluation of the current template and uses a file instead. The current section is executed then the new document is used.

The new document is processed in the same dataspace. The previously defined variables are not destroyed.

Output parameters

- 0: Normal execution.
- Other than zero: Error code (accessible via the `AmwErrorMsg` function.

Example

```
<script Language="AmScript1.0">
iRc = AmwFetchQuery("invalid query",0,10)
if iRc <>0 then
  rem error occurred use standard error reporting doc
  iRc = AmwLoadFile("error.htm")
end if
</script>
```

AmwLogin()

API syntax

```
long AmwLogin(char *strDataBase, char *strUserName, char *strPassword);
```

Internal BASIC syntax

```
Function AmwLogin(strDataBase As String, strUserName As String,
strPassword As String) As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	
Wizard script	
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Connects to an AssetCenter database. AssetCenter Web uses the same connection information as AssetCenter.

Input parameters

- `strDataBase`: Database name (as given in the database connection dialog box in AssetCenter).
- `strUserName`: Login used for the user connection. User login name (SQL name: UserLogin).
- `strPassword`: Password associated with the login. Password (SQL name: lPasswordId).

Output parameters

- 0: Normal execution.
- Other than zero: Error code (accessible via the `AmwErrorMsg` function.

Example

```
iRc = AmwLogin(DbName,User,Password)
```

If iRc <> 0 then Print "Login failed: ";AmwErrorMsg()

AmwLogout()

API syntax

```
long AmwLogout();
```

Internal BASIC syntax

```
Function AmwLogout() As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	
Wizard script	
FINISH.DO script of a wizard	

	Available
AssetCenter Web APIs	X

Description

Disconnects from the current database.

This operation ends the current user work session. No other scripting operation should be done after this call.

This function is generally used as the last line of a script section in the logout template.

Output parameters

- 0: Normal execution.
- Other than zero: Error code (accessible via the `AmwErrorMsg` function.

Example

```
<script Language="AmScript1.0">
 iRc = AmwLogout()
</script>

<H1> Thank you for using this software </H1>
```

AmwNewRecord()

API syntax

```
long AmwNewRecord(char *strTable, char *strFieldList);
```

Internal BASIC syntax

```
Function AmwNewRecord(strTable As String, strFieldList As String) As
Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	
Wizard script	
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Creates a empty record in an AssetCenter table identified by its SQL name then uses the `strFieldList` parameter to define the initial values of the record fields.

The initial values of fields are read from the variables specified in the `strFieldList` parameter. The `strFieldList` parameter has the following syntax:

```
<SQL name of field 1>(<Variable name>);<SQL name of field
2>(<Variable name>)...
```

Input parameters

- `strTable`: SQL name of the table.
- `strFieldList`: tuple of values <SQL name of field>(<Name of variable>).

Output parameters

- 0: Normal execution.
- Other than zero: Error code (accessible via the `AmwErrorMsg` function.

Example

```
AmwSetTpl "Telephone","12121212"
```

AmwSetTpl "UserName","joe"

iRc = AmNewRecord("amEmplDeptId","Name(UserName);Tel(Telephone)")

AmWorkTimeSpanBetween()

API syntax

```
long AmWorkTimeSpanBetween(char *strCalendarSqlName, long tmEnd, long
tmStart);
```

Internal BASIC syntax

```
Function AmWorkTimeSpanBetween(strCalendarSqlName As String, tmEnd As
Date, tmStart As Date) As Date
```

Field of application

Version: 3.00

	Available
Built-in function	
AssetCenter APIs	X
Configuration script of a field or link	X
Definition script of a feature parameter	X

	Available
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns the duration of working periods between two dates. This duration is expressed in seconds; it respects the information in a calendar of working periods.

Input parameters

- `strCalendarSqlName`: This parameter contains the SQL name of the calendar of working periods used to calculate the duration of the working period between the two dates. If this parameter is omitted, the calculated duration does not take working periods into account.
- `tmEnd`: This parameter contains the end date for the period used in calculating the working period.
- `tmStart`: This parameter contains the start date for the period used in calculating the working period.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Example

The following example calculates the working period between 01/09/1998 at 8 a.m. and 24/09/1998 at 7 p.m. The calendar used, whose SQL name is "Calendar_Paris", defines the following working periods:

- From Monday to Thursday from 8 a.m. to 12 noon, then from 2 p.m. to 6 p.m.
- Fridays from 8 a.m. to 12 noon, then from 2 p.m. to 5 p.m.

```
AmWorkTimeSpanBetween("Calendar_Paris", "1998/09/24 19:00:00",
"1998/09/01 08:00:00")
```

This example returns the value 507,600 which represents the number of working seconds between the two dates.

AmwOutputIf()

API syntax

```
long AmwOutputIf();
```

Internal BASIC syntax

```
Function AmwOutputIf() As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	

	Available
Wizard script	
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Allows generation of the current template depending on the value of a variable identified by its name.

Generation is possible if the variable exists and is not zero. Otherwise, the following lines of codes are ignored until another `AmwOutoutIf` function is executed with success.

This function can be used to compose HTML documents containing information to generate in case of error or failure.

For example, on execution, the script section defines the value of a variable if the operation is successfully executed. This variable is then used to determine which part of the HTML document is to be displayed.

Output parameters

- 0: Normal execution.
- Other than zero: Error code (accessible via the `AmwErrorMsg` function.

Example

```
<script Language="AmScript1.0">
 rem set one valid var
 AmwSetTpl "GoodVar","true"
 rem set one false var
 AmwSetTpl "FalseVar","False"
</script>
$$(AmwOutputIf("NoVarsLikeThis"))
This line is not visible because var does not exist<br>
```

$$(AmwOutputIf("FalseVar"))

This line is not visible because var evaluates as false

$$(AmwOutputIf("GoodVar"))

This line is visible because this var evaluates as true

AmwOutputIfNot()

API syntax

```
long AmwOutputIfNot();
```

Internal BASIC syntax

```
Function AmwOutputIfNot() As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	
Wizard script	
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Allows generation of the current template depending on the value of a variable identified by its name. This function is the exact opposite of the `AmwOutputIf` function. Generation is if the variable does not exist or is zero.

Output parameters

- 0: Normal execution.
- Other than zero: Error code (accessible via the `AmwErrorMsg` function.

AmwPlusToSpace()

API syntax

```
long AmwPlusToSpace(char *return, long lreturn, char *strVal);
```

Internal BASIC syntax

```
Function AmwPlusToSpace(strVal As String) As String
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	

	Available
Wizard script	
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Decodes a CGI type string to a character string that can be displayed.

This function can be used to manually decode the strings provided by the CGI mechanisms.

Input parameters

- `strVal`: Character string to decode.

Output parameters

- Decoded character string.
- An error code is accessible via the `AmwErrorMsg` function.

Example

```
<script Language="amScript1.0">
 varText = "%3F+Cgi+like+string+%3F"
 AmwSetTpl "param1",AmwPlusToSpace(varText)
</script>
```

Following line should display: ? cgi like string ?

$$(Param1)

AmwPrintDebug()

API syntax

```
void AmwPrintDebug(char *strVal);
```

Internal BASIC syntax

```
Function AmwPrintDebug(strVal As String)
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	
Wizard script	
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Displays a string in the local console window. This function can only be used if the executable version of AssetCenter Web is running.

When AssetCenter Web is running as a service, the debug information can only be sent to the connected user. If `amw3.exe` is started manually by

double-clicking the program icon, a console window is created, which displays the additional information.

This function avoids cluttering generated HTML documents with debug information.

Input parameters

- *strVal*: Message to be displayed.

Output parameters

- 0: Normal execution.
- Other than zero: Error code (accessible via the `AmwErrorMsg` function.

Example

```
<script Language="AmScript1.0">
 Print "This text goes to the client"
 AmwPrintDebug "This Text goes in the local console window"
</script>
```

AmwQStrDelKey()

API syntax

```
long AmwQStrDelKey(char *return, long lreturn, char *strQStr, char
*strKey);
```

Internal BASIC syntax

```
Function AmwQStrDelKey(strQStr As String, strKey As String) As String
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	
Wizard script	
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Removes a specific key entry in a CGI query string.

This function allows easy manipulation of query strings and avoids manual parsing.

Input parameters

- `strQStr`: Original query as a character string.
- `strKey`: Name of the key to delete.

Output parameters

- Updated query string.
- An error code is accessible via the `AmwErrorMsg` function.

Example

```
<script Language="AmScript1.0">
```

rem get current query string and remove name parameter

AmwSetTpl "NewQueryStr",AmwQStrDelKey(T_Query,"Name")
</script>

<A HREF="doc.htm?$$(NewQueryStr)"> Submit with no name </A>

AmwQStrSetKey()

API syntax

```
long AmwQStrSetKey(char *return, long lreturn, char *strQStr, char
*strKey, char *strVal);
```

Internal BASIC syntax

```
Function AmwQStrSetKey(strQStr As String, strKey As String, strVal As
String) As String
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	
Wizard script	
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Changes or creates key/value pair in a CGI query string.

Input parameters

- *strQStr*: Original query as a character string.
- *strKey*: Name of the key to modify or to add.
- *strVal*: Value of the key.

Output parameters

- Updated query string.
- An error code is accessible via the `AmwErrorMsg` function.

Example

```
<script Language="AmScript1.0">
 rem get current query string an change name parameter
 AmwSetTpl "NewQueryStr",AmwQStrSetKey(T_Query,"Name","joe")
</script>

<A HREF="doc.htm?$$(NewQueryStr)"> Submit with joe name </A>
```

AmwRefuseRequest()

API syntax

```
long AmwRefuseRequest(long lReqId, char *strComment);
```

Internal BASIC syntax

```
Function AmwRefuseRequest(lReqId As Long, strComment As String) As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	
Wizard script	
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Refuses a purchase request identified by its `lReqId` identifier.

Input parameters

- `lReqId`: Identifier of the purchase request ("amReqApprLine" table).
- `strComment`: Approver's comments.

Output parameters

- 0: Normal execution.
- Other than zero: Error code (accessible via the `AmwErrorMsg` function.

AmwSetEnv()

API syntax

```
void AmwSetEnv(char *strKeyName, char *strValue);
```

Internal BASIC syntax

```
Function AmwSetEnv(strKeyName As String, strValue As String)
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	
Wizard script	
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Sets or creates a variable with a specific value in the global session variable-space.

Input parameters

- *strKeyName*: Name of the variable.
- *strValue*: Value of the variable as a string.

Output parameters

An error code is accessible via the `AmwErrorMsg` function.

Example

```
<script Language="AmScript1.0">
 rem Set a variable for the whole session
 AmwSetEnv "MyRealName","Joe"
</script>
```

AmwSetTpl()

API syntax

```
void AmwSetTpl();
```

Internal BASIC syntax

```
Function AmwSetTpl()
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	

	Available
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	
Wizard script	
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Defines or creates a variable with a value defined in the variable space of the local template.

Output parameters

An error code is accessible via the `AmwErrorMsg` function.

Example

```
<script Language="AmScript1.0">
 rem Set a variable local to this template and visible in html
 AmwSetEnv "StockValue","35$"
</script>

Value: $$(StockValue)<br>
```

AmwSetWebAdminPwd()

API syntax

```
long AmwSetWebAdminPwd(char *strNew, char *strOld);
```

Internal BASIC syntax

```
Function AmwSetWebAdminPwd(strNew As String, strOld As String) As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	
Wizard script	
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Changes the web administrator password.

Input parameters

- `strNew`: New password.

- *strOld*: Old password.

Output parameters

- 0: Normal execution.
- Other than zero: Error code (accessible via the `AmwErrorMsg` function.

Example

```
<script Language="AmScript1.0">
 rem Change password in admin session
 iRc =   AmwSetWebAdminPwd("","K7TR89M")
</script>
```

AmwSetWebTimeout()

API syntax

```
long AmwSetWebTimeout(long iTimeout);
```

Internal BASIC syntax

```
Function AmwSetWebTimeout(iTimeout As Long) As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	
Configuration script of a field or link	

	Available
Definition script of a feature parameter	
"Script" type action	
Wizard script	
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Changes the default timeout value for user sessions defined in the `amw3.ini` file.

If no activity is recorded for *`lTimeOut`* minutes, the user is disconnected.

Input parameters

- *`iTimeout`*: Number of minutes before disconnection of an inactive user.

Output parameters

- 0: Normal execution.
- Other than zero: Error code (accessible via the `AmwErrorMsg` function.

Example

```
<script Language="AmScript1.0">
 rem Change timeout value to 3 mn
 iRc =  AmwSetWebTimeOut(3)
</script>
```

AmwSlaveList()

API syntax

```
long AmwSlaveList();
```

Internal BASIC syntax

```
Function AmwSlaveList() As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	
Wizard script	
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Displays the list of active connections.

Output parameters

- 0: Normal execution.

- Other than zero: Error code (accessible via the `AmwErrorMsg` function.

Example

```
<H1> Active users:</H1>
```

$$(AmwSlaveList())

AmwSpaceToPlus()

API syntax

```
long AmwSpaceToPlus(char *return, long lreturn, char *strVal);
```

Internal BASIC syntax

```
Function AmwSpaceToPlus(strVal As String) As String
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	
Wizard script	
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Encodes an ASCII character string in order to make it compatible with CGI type strings.

This function is useful to create links when additional information must be sent to the destination template.

Input parameters

- *strVal*: Character string to encode.

Output parameters

- Encoded string.
- An error code is accessible via the `AmwErrorMsg` function.

Example

```
<script Language="amScript1.0">
 varText = "Name with strange chars +=$? "
 AmwSetTpl "param1",AmwPlusToSpace(varText)
</script>

<A HREF="doc.htm?$$Value=$$(param1)> Click me </A>
```

AmwStrReplace()

API syntax

```
long AmwStrReplace(char *return, long lreturn, char *strString, char
*strFind, char *strRepl);
```

Internal BASIC syntax

```
Function AmwStrReplace(strString As String, strFind As String, strRepl
As String) As String
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	
Wizard script	
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Replaces all occurrences of the string `strFind` in the string `strString` by the string `strRepl`.

Input parameters

- `strString`: Character string to process.
- `strFind`: String to bChaîne à remplacer.
- `strRepl`: Replacement string.

Output parameters

- String after replacement.
- The error message can be accessed via the `AmwErrorMsg` function.

Example

```
<script Language="AmScript1.0">
  rem replace all occurrences of joe with jack
  resStr = AmwStrReplace("Default user is joe","joe","jack")
</script>
```

AmwUpdateRecord()

API syntax

```
long AmwUpdateRecord(char *strTable, char *strFieldList, long lId);
```

Internal BASIC syntax

```
Function AmwUpdateRecord(strTable As String, strFieldList As String,
lId As Long) As Long
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	
Wizard script	
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Updates the fields of a record identified by its identifier and the SQL name of the table in which it is found.

New values for fields are read from variables specified in the *strFieldList* parameter. The *strFieldList* parameter has the following syntax:

```
<SQL name of field 1>(<Variable name>);<SQL name of field
2>(<Variable name>)...
```

The fields are updated with the value contained in the variable specified between brackets.

Input parameters

- *strTable*: Name of the table containing the record to be modified.
- *strFieldList*: tuple of values <SQL name of field>(<Name of variable>).
- *lId*: Identifier of the record.

Output parameters

- 0: Normal execution.
- Other than zero: Error code (accessible via the `AmwErrorMsg` function.

Example

```
<script Language="AmScript1.0">
AmwSetTpl "NewTel","12121212"
AmwSetTpl "NewName","joe"
iRc=AmUpdateRecord("amEmplDeptId","Name(NewName);Tel(NewTel)"
,lId)
</script>
```

AmwVars()

API syntax

```
void AmwVars();
```

Internal BASIC syntax

```
Function AmwVars()
```

Field of application

Version: 2.52

	Available
Built-in function	
AssetCenter APIs	
Configuration script of a field or link	
Definition script of a feature parameter	
"Script" type action	
Wizard script	
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Displays the contents of the local and global variable in the current document.

Output parameters

An error code is accessible via the `AmwErrorMsg` function.

Example

```
<H1>Available variables:</H1>
```

$$(AmwVars())

AppendOperand()

Internal BASIC syntax

```
Function AppendOperand(strExpr As String, strOperator As String,
strOperand As String) As String
```

Field of application

Version: 3.5

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Concatenates a string according to the parameters passed to the function. The results are given as follows:

strExpr strOperator strOperand

Input parameters

- *strExpr*: Expression to be concatenated.
- *strOperator*: Operator to concatenate.
- *strOperand*: Operand to concatenate.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Notes

If one of the *strExpr* or *strOperand* parameters is omitted, *strOperator* is not used in the concatenation.

ApplyNewVals()

Internal BASIC syntax

```
Function ApplyNewVals(strValues As String, strNewVals As String,
strRows As String, strRowFormat As String) As String
```

Field of application

Version: 3.5

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Assigns identical values to identical cells in a "ListBox" control.

Input parameters

- `strValues`: Source string containing the values of a "ListBox" control to be processed.
- `strNewVals`: New value to assign to the cells concerned.
- `strRows`: Identifiers of lines to be processed. The identifiers are separated by commas.
- `strRowFormat`: Formatting instructions for the sublist. Instructions are separated by the "|" character. Each instruction represents the number of the column containing the `strNewVals` parameter.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.

- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Asc()

Internal BASIC syntax

```
Function Asc(strAsc As String) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Returns a numeric value that is the ASCII code for the first character in a string.

Input parameters

- *strAsc*: Character sting on which the function operates.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

Example

```
Sub Main ()
   Dim I, Msg                    ' Declare variables.
   For I = Asc("A") To Asc("Z")    ' From A through Z.
   Msg = Msg & Chr(I)          ' Create a string.
   Next I
   Print Msg                    ' Display results.
End Sub
```

Atn()

Internal BASIC syntax

```
Function Atn(dValue As Double) As Double
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X

	Available
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Returns the arc tangent of a number, expressed in radians.

Input parameters

- `dValue`: Number for which you want to know the arc tangent.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Example

```
Sub AtnExample ()
   Dim Msg, Pi        ' Declare variables.
   Pi = 4 * Atn(1)    ' Calculate Pi.
   Msg = "Pi is equal to " & Str(Pi)
   Print Msg          ' Display results.
End Sub
```

BasicToLocalDate()

Internal BASIC syntax

```
Function BasicToLocalDate(strDateBasic As String) As String
```

Field of application

Version: 3.5

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function converts a Basic format date to a string format date (as displayed in Windows Control Panel).

Input parameters

- `strDateBasic`: Date in Basic format to convert.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

BasicToLocalTime()

Internal BASIC syntax

```
Function BasicToLocalTime(strTimeBasic As String) As String
```

Field of application

Version: 3.5

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function converts a Basic format time to a string format time (as displayed in Windows Control Panel).

Input parameters

- *strTimeBasic*: Time in Basic format to convert.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

BasicToLocalTimeStamp()

Internal BASIC syntax

```
Function BasicToLocalTimeStamp(strTSBasic As String) As String
```

Field of application

Version: 3.5

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function converts a Date+Time in Basic format to a Date+Time in string format (as displayed in Windows Control Panel).

Input parameters

- *`strTSBasic`*: Date+Time in Basic format to convert.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Beep()

Internal BASIC syntax

```
Function Beep()
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X

	Available
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Plays a beep on the machine.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

CDbl()

Internal BASIC syntax

```
Function CDbl(dValue As Double) As Double
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	

	Available
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Converts an expression to a "Double".

Input parameters

- `dValue`: Expression to convert.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Example

```
Sub Main ()
  Dim y As Long

  y = 25
  If  VarType(y) = 2 Then
  Print y
```

```
    x = CDbl(y)   'Converts the long value of y to an integer value in x
      Print x
    End If
End Sub
```

ChDir()

Internal BASIC syntax

```
Function ChDir(strDirectory As String)
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Changes the current directory.

Input parameters

- *strDirectory*: New current directory.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

ChDrive()

Internal BASIC syntax

```
Function ChDrive(strDrive As String)
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Changes the current drive.

Input parameters

- `strDrive`: New drivename.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Chr()

Internal BASIC syntax

```
Function Chr(iChr As Long) As String
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X

	Available
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Returns a string corresponding to the ASCII passed by the `iChr` parameter.

Input parameters

- `iChr`: ASCII code of the character.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Example

```
Sub ChrExample ()
  Dim X, Y, Msg, NL
  NL = Chr(10)
  For X = 1 to 2
    For Y = Asc("A") To Asc("Z")
      Msg = Msg & Chr(Y)
    Next Y
  Msg = Msg & NL
  Next X
  Print Msg
```

End Sub

CInt()

Internal BASIC syntax

```
Function CInt(iValue As Long) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Converts any valid expression to an integer.

Input parameters

- `iValue`: Expression to convert.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

Example

```
Sub Main ()
   Dim y As Long

   y = 25
   If  VarType(y) = 2 Then
   Print y
   x = CInt(y)   'Converts the long value of y to an integer value in x
      Print x
   End If
End Sub
```

CLng()

Internal BASIC syntax

```
Function CLng(lValue As Long) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X

	Available
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Converts any valid expression into a long.

Input parameters

- `lValue`: Expression to convert.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Example

```
Sub Main ()
  Dim y As Integer

  y = 25
  If  VarType(y) = 2 Then
  Print y
  x = CLng(y)   'Converts the integer value of y to a long value in x
```

```
    Print x
  End If
End Sub
```

Cos()

Internal BASIC syntax

```
Function Cos(dValue As Double) As Double
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Returns the cosine of a number, expressed in radians.

Input parameters

- `dValue`: Number whose cosine you want to know.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

CountOccurences()

Internal BASIC syntax

```
Function CountOccurences(strSearched As String, strPattern As String,
strEscChar As String) As Long
```

Field of application

Version: 3.5

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Counts the number of occurrences of a string inside another string.

Input parameters

- *strSearched*: Character string in which to perform to the search.
- *strPattern*: Character string to find inside the *strSearched* parameter.
- *strEscChar*: Escape character. If the function encounters this character inside the *strSearched* string, the search stops.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Example

```
Dim MyStr
```

MyStr=CountOccurences("you | me | you,me | you", "you", ",") 'Returns "2"

MyStr=CountOccurences("you | me | you,me | you", "you", " | ") 'Returns "1"

CountValues()

Internal BASIC syntax

```
Function CountValues(strSearched As String, strSeparator As String,
strEscChar As String) As Long
```

Field of application

Version: 3.5

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Counts the number of elements in a string, taking into account a separator and an escape character.

Input parameters

- `strSearched`: Character string to process.
- `strSeparator`: Separator used to delimit the elements.
- `strEscChar`: Escape character. If this character prefixes a separator, it will be ignored.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Example

```
Dim MyStr
MyStr=CountValues("you|me|you\|me|you", "|", "\") 'Returns 4
MyStr=CountValues("you|me|you\|me|you", "|", "") 'Returns 5
```

CSng()

Internal BASIC syntax

```
Function CSng(fValue As Single) As Single
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Converts any valid expression to a floating point number ("Float").

Input parameters

- `fValue`: Expression to convert.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

CStr()

Internal BASIC syntax

```
Function CStr(strValue As String) As String
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Converts any valid expression to a String.

Input parameters

- *strValue*: Expression to convert.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

CurDir()

Internal BASIC syntax

```
Function CurDir() As String
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X

	Available
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Returns the current path.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

CVar()

Internal BASIC syntax

```
Function CVar(vValue As Variant) As Variant
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X

	Available
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Converts any valid expression to a Variant.

Input parameters

- `vValue`: Expression to convert.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Date()

Internal BASIC syntax

```
Function Date() As Date
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Returns the current system date

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

DateSerial()

Internal BASIC syntax

```
Function DateSerial(iYear As Long, iMonth As Long, iDay As Long) As
Date
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns a date formatted according to the `iYear`, `iMonth` and `iDay` parameters.

Input parameters

- `iYear`: Year. If the value is between 0 and 99, this parameter describes the years from 1900 to 1999. For all other years, you must specify the four figures (e.g. 1800).
- `iMonth`: Month.

- *iDay*: Day.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Example

Each of these parameters can be set to a numeric expression representing a number of days, months or years. For example:

```
DateSerial(1999-10, 3-2, 15-8)
```

Returns the value:

```
1989/1/7
```

When the value of a parameter is out of the expected range (i.e. 1-31 for days, 1-12 for months, etc.), it is converted to the parameter the next up. Thus, if you enter "35" for the *iDay* parameter, it will be interpreted as 1 month and 4 days.

The following example:

```
DateSerial (1999-50, 9-5, 1-2)
```

Returns the value:

```
1949/3/30
```

DateValue()

Internal BASIC syntax

```
Function DateValue(tmDate As Date) As Date
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns the date portions of a "Date+Time" value.

Input parameters

- `tmDate`: "Date+Time" format date.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Example

The following example:

```
DateValue ("1999/09/24 15:00:00")
```

Returns the value:

```
1999/09/24
```

Day()

Internal BASIC syntax

```
Function Day(tmDate As Date) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Returns the day contained in the `tmDate` parameter.

Input parameters

- `tmDate`: Parameter in Date+Time format to be processed.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

Example

```
Dim MyStr
```

MyStr=Day(AmDate())'Returns the current day

EscapeSeparators()

Internal BASIC syntax

```
Function EscapeSeparators(strSource As String, strSeparators As String,
strEscChar As String) As String
```

Field of application

Version: 3.5

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Prefixes one or more separator characters with an escape character.

Input parameters

- *strSource*: Character string to process.
- *strSeparators*: List of separators to be prefixed. If you want to declare several separators, you must separate them with the escape character (indicated in the *strEscChar* parameter.
- *strEscChar*: Escape character. It will be used to prefix all separators in *strSeparators*.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Example

```
Dim MyStr
```

MyStr=EscapeSeparators("you|me|you,me|you", "|\,", ",") 'Returns "you\|me\|you\,me\|you"

Exp()

Internal BASIC syntax

```
Function Exp(dValue As Double) As Double
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Returns the exponent of a number.

Input parameters

- `dValue`: Number whose exponent you want to know.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

ExtractValue()

Internal BASIC syntax

```
Function ExtractValue(strSeparator As String, strEscChar As String) As
String
```

Field of application

Version: 3.5

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Extracts from a string the values delimited by a separator. The recovered value is deleted from the source string. This operation takes into account a possible escape character. If the separator is not found in the source string, the whole string is returned and the source string is deleted in full.

Input parameters

- `strSeparator`: Character used as separator in the source string.

- *strEscChar*: Escape character. If this character prefixes the separator, it will be ignored.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Example

```
Dim MyStr
```

MyStr=ExtractValue("you,me", ",", "\") 'Returns "you" and leaves "me" in the source string

MyStr=ExtractValue(",you,me", ",", "\") 'Returns "" and leaves "you,me" in the source string

MyStr=ExtractValue("you", ",", "\") 'Returns "you" and leaves "" in the source string

MyStr=ExtractValue("you\,me", ",", "\") 'Returns "you\,me" and leaves "" in the source string

MyStr=ExtractValue("you\,me", ",", "") 'Returns "you\" and leaves "me" in the source string

FileCopy()

Internal BASIC syntax

```
Function FileCopy(strSource As String, strDest As String) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Copies a file or a folder.

Input parameters

- `strSource`: Full path of the file or directory to copy.
- `strDest`: Full path of the target file or directory.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

FileDateTime()

Internal BASIC syntax

```
Function FileDateTime(strFileName As String) As Date
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Returns the time and date of a file as a Long.

Input parameters

- `strFileName`: Full path name of the file concerned by the operation.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

FileLen()

Internal BASIC syntax

```
Function FileLen(strFileName As String) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Returns the size of a file.

Input parameters

- `strFileName`: Full path name of the file concerned by the operation.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Fix()

Internal BASIC syntax

```
Function Fix(dValue As Double) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Returns the integer portion of a number (first greatest integer in the case of a negative number).

Input parameters

- `dValue`: Number whose integer portion you want to know.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Format()

Internal BASIC syntax

```
Function Format(vValue As Variant, strFormat As String) As String
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Formats a number according to the expression contained in the *strFormat* parameter.

Input parameters

- *vValue*: Number to be formatted.
- *strFormat*: Expression containing the formatting instructions.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Example

The following example of code show how to format number:

```
Dim MyValue

MyValue=3245.69

MyStr=FormatDate(MyValue, "###0.000") 'Returns "Tuesday 14
March 2000"
```

FormatDate()

Internal BASIC syntax

```
Function FormatDate(tmFormat As Date, strFormat As String) As String
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Formats a date according to the expression contained in the `strFormat` parameter.

Input parameters

- `tmFormat`: Date to be formatted.
- `strFormat`: Expression containing the formatting instructions.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Example

The following example of code shows how to format a date:

```
Dim MyDate
```

```
MyDate="2000/03/14"

MyStr=FormatDate(MyDate, "dddd d mmmm yyyy") 'Returns "Tuesday
14 March 2000"
```

FormatResString()

Internal BASIC syntax

```
Function FormatResString(strResString As String, strParamOne As String,
strParamTwo As String, strParamThree As String, strParamFour As String,
strParamFive As String) As String
```

Field of application

Version: 3.5

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function processes a source string, replacing the variable $1, $2, $3, $4, and $5 with the strings passed in the `strParamOne`, `strParamTwo`, `strParamThree`, `strParamFour`, and `strParamFive` parameters.

Input parameters

- *strResString*: Source string to be processed.
- *strParamOne*: Replacement string of variable $1.
- *strParamTwo*: Replacement string of variable $2.
- *strParamThree*: Replacement string of variable $3.
- *strParamFour*: Replacement string of variable $4.
- *strParamFive*: Replacement string of variable $5.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Example

The following example:

```
FormatResString("I$1he$2you$3", "you", "we", "they")
```

returns "Iyouheweyouthey".

FormatString()

Internal BASIC syntax

```
Function FormatString(strValue As String, strFormat As String) As
String
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Formats a string according to the expression contained in the `strFormat` parameter.

Input parameters

- `strValue`: Character string to format
- `strFormat`: Expression containing the formatting instructions.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Example

The following example of code show how to format a character string:

```
Dim MyString
```

```
MyString="2000/03/14"
MyStr=FormatDate(MyDate, "dddd d mmmm yyyy") 'Returns "Tuesday
14 March 2000"
```

FV()

Internal BASIC syntax

```
Function FV(dblRate As Double, iNper As Long, dblPmt As Double, dblPV
As Double, iType As Long) As Double
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns the future amount of an annuity based on constant and periodic payments, with a set interest rate.

Input parameters

- *dblRate*: This parameter indicates the interest rate per date of payment. For example, the rate per date of payment for a loan with a 6% annual interest rate, paid back by monthly dates of payment, would be:

```
0.06/12=0.005 or 0.5%
```

- *iNper*: This parameter contains the total number of dates of payment for the financial operation.
- *dblPmt*: This parameter indicates the amount of the payment to be made at each date of payment. The payment generally includes both principal and interest.
- *dblPV*: This parameter contains the actual value (or overall sum) for a series of payments to be made in the future.
- *iType*: This parameter indicates the payment deadline. It can have one of the following values:
  - *0* if the payments are due in arrears (i.e. at the end of the period)
  - *1* if the payments are due in advance (i.e. at the start of the period)

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Notes

- The *Rate* and *Nper* parameters must be calculated using payments expressed in the same units.
- Amounts paid (expressed in particular by the *Pmt* parameter) are represented by negative numbers. Sums received are represented by positive numbers.

GetListItem()

Internal BASIC syntax

```
Function GetListItem(strFrom As String, strSep As String, lNb As Long,
strEscChar As String) As String
```

Field of application

Version: 3.5

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Returns the `lNb`th portion of a string delimited by separators.

Input parameters

- `strFrom`: Source string to be processed.
- `strSep`: Character used as separator in the source string.
- `lNb`: Position of the string to recover.

- *strEscChar*: Escape character. If this character prefixes a separator, it will be ignored.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Example

The following example:

```
GetListItem("this_is_a_test", "_", 2, "%")
```

returns "is".

```
GetListItem("this%_is_a_test", "_", 2, "%")
```

returns "a".

Hex()

Internal BASIC syntax

```
Function Hex(dValue As Double) As String
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	

	Available
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Returns the hexadecimal value of a decimal parameter.

Input parameters

- `dValue`: Decimal number whose hexadecimal value you want to know.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Hour()

Internal BASIC syntax

```
Function Hour(tmTime As Date) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Returns the hour value contained in the `tmTime` parameter.

Input parameters

- `tmTime`: Parameter in Date+Time format to be processed.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

Example

```
Dim MyStr
```

MyStr=Hour(AmDate())'Returns the current hour, for example "15" if the time is 15:45:30

InStr()

Internal BASIC syntax

```
Function InStr(iPosition As Long, strSource As String, strPattern As
String) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Returns the character position of the first occurrence of a string within a string.

Input parameters

- `iPosition`: Starting point of the search. This parameter is not optional and must be a valid positive integer no greater than 65,535.
- `strSource`: String in which the search is performed.

- *strPattern*: String to search.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

Example

```
Sub Main ()
   B$ = "Good Bye"
   A% = InStr(2, B$, "Bye")
   C% = Instr(3, B$, "Bye")
End Sub
```

Int()

Internal BASIC syntax

```
Function Int(dValue As Double) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X

	Available
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Returns the integer portion of a number (first lesser than integer in the case of a negative number).

Input parameters

- `dValue`: Number whose integer portion you want to know.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

IPMT()

Internal BASIC syntax

```
Function IPMT(dblRate As Double, iPer As Long, iNper As Long, dblPV As
Double, dblFV As Double, iType As Long) As Double
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns the amount of interest for an given date of payment of an annuity.

Input parameters

- `dblRate`: This parameter indicates the interest rate per date of payment. For example, the rate per date of payment for a loan with a 6% annual interest rate, paid back by monthly dates of payment, would be:

  ```
  0.06/12=0.005 or 0.5%
  ```

- `iPer`: This parameter indicates the period for the calculation, between 1 and the value of the `Nper` parameter.
- `iNper`: This parameter contains the total number of dates of payment for the financial operation.
- `dblPV`: This parameter contains the actual value (or overall sum) for a series of payments to be made in the future.
- `dblFV`: This parameter contains the future value or the balance that you want to obtain after having paid the final date of payment. In general, and particularly when reimbursing a loan, this parameter is set to "0". In effect, once you have made all the dates of payment, the value of the loan is nil.

- *iType*: This parameter indicates the payment deadline. It can have one of the following values:
  - *0* if the payments are due in arrears (i.e. at the end of the period)
  - *1* if the payments are due in advance (i.e. at the start of the period)

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Notes

- The *Rate* and *Nper* parameters must be calculated using payments expressed in the same units.
- Amounts paid (expressed in particular by the *Pmt* parameter) are represented by negative numbers. Sums received are represented by positive numbers.

Kill()

Internal BASIC syntax

```
Function Kill(strKilledFile As String) As Long
```

Field of application

Version: 3.00

	Available

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Deletes a file.

Input parameters

- `strKilledFile`: Full path of the file concerned by the operation.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

LCase()

Internal BASIC syntax

```
Function LCase(strString As String) As String
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Returns a string in which all letters of the string parameter have been converted to lower case.

Input parameters

- `strString`: Character string to convert to lowercase.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Example

```
' This example uses the LTrim and RTrim functions to strip
leading ' and trailing spaces, respectively, from a string
variable.
' It uses the Trim function alone to strip both types of spaces.
' LCase and UCase are also shown in this example as well as the use
' of nested function calls

Sub Main
   MyString = "  <-Trim->  "    ' Initialize string.
   TrimString = LTrim(MyString)    ' TrimString = "<-Trim->  ".
   Print "|" & TrimString & "|"
   TrimString = LCase(RTrim(MyString))    ' TrimString = "  <-trim->".
   Print "|" & TrimString & "|"
   TrimString = LTrim(RTrim(MyString))    ' TrimString = "<-Trim->".
   Print "|" & TrimString & "|"
   ' Using the Trim function alone achieves the same result.
   TrimString = UCase(Trim(MyString))    ' TrimString = "<-TRIM->".
   Print "|" & TrimString & "|"
End Sub
```

Left()

Internal BASIC syntax

```
Function Left(strString As String, iNumber As Long) As String
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Returns the left most iNumber characters of a string parameter.

Input parameters

- *`strString`*: Character string to process.
- *`iNumber`*: Number of characters to return.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Example

```
Sub Main ()
  Dim LWord, Msg, RWord, SpcPos, UsrInp    ' Declare variables.
  Msg = "Enter two words separated by a space."
```

```
    UsrInp = InputBox(Msg)    ' Get user input.
    print UsrInp
    SpcPos = InStr(1, UsrInp, " ")    ' Find space.
    If SpcPos Then
        LWord = Left(UsrInp, SpcPos - 1)    ' Get left word.
        print "LWord: ";  LWord
        RWord = Right(UsrInp, Len(UsrInp) - SpcPos)    ' Get right word.
        Msg = "The first word you entered is " & LWord
        Msg = Msg & "." & " The second word is "
        Msg = "The first word you entered is <" & LWord & ">"
        Msg = Msg & RWord & "."
    Else
        Msg = "You didn't enter two words."
    End If
    Print Msg    ' Display message.
    MidTest = Mid("Mid Word Test", 4, 5)
    Print MidTest
End Sub
```

LeftPart()

Internal BASIC syntax

```
Function LeftPart(strFrom As String, strSep As String, bCaseSensitive
As Long) As String
```

Field of application

Version: 3.5

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Extracts the portion of a string to the left of the separator specified in the `strSep` parameter.

The search for the separator is performed from left to right.

The search can be made case sensitive using the `bCaseSensitive` parameter.

Input parameters

- `strFrom`: Source string to be processed.
- `strSep`: Character used as separator in the source string.
- `bCaseSensitive`: Depending on this parameter, the search is case sensitive (=1) or not (=0).

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Example

These examples illustrate use of the `LeftPart`, `LeftPartFromRight`, `RightPart`, and `RightPartFromLeft` functions on the same string: "This_is_a_test":

```
LeftPart("This_is_a_test","_",0)
```

Returns "This".

```
LeftPartFromRight("This_is_a_test","_",0)
```

Returns "This_is_a".

```
RightPart("This_is_a_test","_",0)
```

Returns "test".

```
RightPartFromLeft("This_is_a_test","_",0)
```

Returns "is_a_test".

LeftPartFromRight()

Internal BASIC syntax

```
Function LeftPartFromRight(strFrom As String, strSep As String,
bCaseSensitive As Long) As String
```

Field of application

Version: 3.5

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X

	Available
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Extracts the portion of a string to the left of the separator specified in the `strSep` parameter.

The search for the separator is performed from right to left.

The search can be made case sensitive using the `bCaseSensitive` parameter.

Input parameters

- `strFrom`: Source string to be processed.
- `strSep`: Character used as separator in the source string.
- `bCaseSensitive`: Depending on this parameter, the search is case sensitive (=1) or not (=0).

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Example

These examples illustrate use of the `LeftPart`, `LeftPartFromRight`, `RightPart`, and `RightPartFromLeft` functions on the same string: "This_is_a_test":

```
LeftPart("This_is_a_test","_",0)
```

Returns "This".

```
LeftPartFromRight("This_is_a_test","_",0)
```

Returns "This_is_a".

```
RightPart("This_is_a_test","_",0)
```

Returns "test".

```
RightPartFromLeft("This_is_a_test","_",0)
```

Returns "is_a_test".

Len()

Internal BASIC syntax

```
Function Len(vValue As Variant) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Returns the number of characters in a string or a variant.

Input parameters

- *vValue*: Variant concerned by the operation.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Example

```
Sub Main ()
   A$ = "Cypress Enable"
   StrLen% = Len(A$)    'the value of StrLen is 14
   Print StrLen%
End Sub
```

LocalToBasicDate()

Internal BASIC syntax

```
Function LocalToBasicDate(strDateLocal As String) As String
```

Field of application

Version: 3.5

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function converts a string format date (as displayed in Windows Control Panel) to a Basic format date .

Input parameters

- `strDateLocal`: Date as string to convert.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

LocalToBasicTime()

Internal BASIC syntax

```
Function LocalToBasicTime(strTimeLocal As String) As String
```

Field of application

Version: 3.5

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function converts a string format time (as displayed in Windows Control Panel) to a Basic format time.

Input parameters

- `strTimeLocal`: Time in string format to convert.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

LocalToBasicTimeStamp()

Internal BASIC syntax

```
Function LocalToBasicTimeStamp(strTSLocal As String) As String
```

Field of application

Version: 3.5

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function converts a Date+Time in string format (as displayed in Windows Control Panel) to a Date+Time in Basic format.

Input parameters

- `strTSLocal`: Date+Time in string format to convert.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Log()

Internal BASIC syntax

```
Function Log(dValue As Double) As Double
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Returns the natural log of a number.

Input parameters

- `dValue`: Number whose logarithm you want to know.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

LTrim()

Internal BASIC syntax

```
Function LTrim(strString As String) As String
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X

	Available
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Removes all leading spaces in a string.

Input parameters

- *`strString`*: Character string to process.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Example

```
' This example uses the LTrim and RTrim functions to strip
leading ' and trailing spaces, respectively, from a string
variable.
```

```
' It uses the Trim function alone to strip both types of spaces.
' LCase and UCase are also shown in this example as well as the use
' of nested function calls

Sub Main
   MyString = "  <-Trim->  "    ' Initialize string.
   TrimString = LTrim(MyString)    ' TrimString = "<-Trim->  ".
   Print "|" & TrimString & "|"
```

```
    TrimString = LCase(RTrim(MyString))    ' TrimString = "  <-trim->".
    Print "|" & TrimString & "|"
    TrimString = LTrim(RTrim(MyString))    ' TrimString = "<-Trim->".
    Print "|" & TrimString & "|"
    ' Using the Trim function alone achieves the same result.
    TrimString = UCase(Trim(MyString))    ' TrimString = "<-TRIM->".
    Print "|" & TrimString & "|"
End Sub
```

MakeInvertBool()

Internal BASIC syntax

```
Function MakeInvertBool(lValue As Long) As Long
```

Field of application

Version: 3.5

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns the an inverse Boolean; (0 becomes 1, all other numbers become 0).

Input parameters

- *lValue*: Number concerned by the operation.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Example

```
Dim MyValue
```

MyValue=MakeInvertBool(0) 'Returns 1

MyValue=MakeInvertBool(1) 'Returns 0

MyValue=MakeInvertBool(254) 'Returns 0

Mid()

Internal BASIC syntax

```
Function Mid(strString As String, iStart As Long, iLen As Long) As
String
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Returns a substring within a string.

Input parameters

- `strString`: String concerned by the operation.
- `iStart`: Start position of the string to extract from within strString.
- `iLen`: Length of the string to extract.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Example

```
Sub Main ()
```

```
    Dim LWord, Msg, RWord, SpcPos, UsrInp    ' Declare variables.
    Msg = "Enter two words separated by a space."
    UsrInp = InputBox(Msg)    ' Get user input.
    print UsrInp
    SpcPos = InStr(1, UsrInp, " ")    ' Find space.
    If SpcPos Then
        LWord = Left(UsrInp, SpcPos - 1)    ' Get left word.
        print "LWord: ";  LWord
        RWord = Right(UsrInp, Len(UsrInp) - SpcPos)    ' Get right word.
        Msg = "The first word you entered is " & LWord
        Msg = Msg & "." & " The second word is "
        Msg = "The first word you entered is <" & LWord & ">"
        Msg = Msg & RWord & "."
    Else
        Msg = "You didn't enter two words."
    End If
    Print Msg    ' Display message.
    MidTest = Mid("Mid Word Test", 4, 5)
    Print MidTest
End Sub
```

Minute()

Internal BASIC syntax

```
Function Minute(tmTime As Date) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Returns the number of minutes contained in the time expressed in the `tmTime` parameter.

Input parameters

- `tmTime`: Parameter in Date+Time format to be processed.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

Example

```
Dim MyStr
```

MyStr=Minute(AmDate())'Returns the number of minutes elapsed in the current hour, for example "45" if the time is 15:45:30

MkDir()

Internal BASIC syntax

```
Function MkDir(strMkDirectory As String) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Creates a new directory.

Input parameters

- `strMkDirectory`: Full path of the directory to create.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

Month()

Internal BASIC syntax

```
Function Month(tmDate As Date) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Returns the month contained in the date expressed in the `tmDate` parameter.

Input parameters

- `tmDate`: Parameter in Date+Time format to be processed.

Output parameters

- 0: Normal execution.

- Other than zero: Error code.

Example

```
Dim MyStr
```

MyStr=Month(AmDate())'Returns the current month

Name()

Internal BASIC syntax

```
Function Name(strSource As String, strDest As String)
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Changes the name of file.

Input parameters

- `strSource`: Full path of the file to rename.
- `strDest`: New file name.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Now()

Internal BASIC syntax

```
Function Now() As Date
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	

	Available
AssetCenter Web APIs	X

Description

Returns the current date and time.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

NPER()

Internal BASIC syntax

```
Function NPER(dblRate As Double, dblPmt As Double, dblPV As Double,
dblFV As Double, iType As Long) As Double
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X

	Available
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns the number of payments of an annuity based on constant and periodic payments, and at a constant interest rate.

Input parameters

- `dblRate`: This parameter indicates the interest rate per date of payment. For example, the rate per date of payment for a loan with a 6% annual interest rate, paid back by monthly dates of payment, would be:

```
0.06/12=0.005 or 0.5%
```

- `dblPmt`: This parameter indicates the amount of the payment to be made at each date of payment. The payment generally includes both principal and interest.
- `dblPV`: This parameter contains the actual value (or overall sum) for a series of payments to be made in the future.
- `dblFV`: This parameter contains the future value or the balance that you want to obtain after having paid the final date of payment. In general, and particularly when reimbursing a loan, this parameter is set to "0". In effect, once you have made all the dates of payment, the value of the loan is nil.
- `iType`: This parameter indicates the payment deadline. It can have one of the following values:
  - *0* if the payments are due in arrears (i.e. at the end of the period)
  - *1* if the payments are due in advance (i.e. at the start of the period)

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Notes

Amounts paid (expressed in particular by the *`Pmt`* parameter) are represented by negative numbers. Sums received are represented by positive numbers.

Oct()

Internal BASIC syntax

```
Function Oct(dValue As Double) As String
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	

	Available
AssetCenter Web APIs	X

Description

Returns the octal value of the decimal parameter.

Input parameters

- `dValue`: Number whose octal value you want to know.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

ParseDMYDate()

Internal BASIC syntax

```
Function ParseDMYDate(strDate As String) As Date
```

Field of application

Version: 3.5

	Available
Built-in function	X

	Available
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns a Date object (as understood in Basic) from a date fomatted as follows:

`dd/mm/yyyy`

Input parameters

- *`strDate`*: Date stored as a string.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

ParseMDYDate()

Internal BASIC syntax

```
Function ParseMDYDate(strDate As String) As Date
```

Field of application

Version: 3.5

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns a Date object (as understood in Basic) from a date fomatted as follows:

`mm/dd/yyyy`

Input parameters

- *`strDate`*: Date stored as a string.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

ParseYMDDate()

Internal BASIC syntax

```
Function ParseYMDDate(strDate As String) As Date
```

Field of application

Version: 3.5

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns a Date object (as understood in Basic) from a date fomatted as follows:

`yyyy/mm/dd`

Input parameters

- *`strDate`*: Date stored as a string.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

PMT()

Internal BASIC syntax

```
Function PMT(dblRate As Double, iNper As Long, dblPV As Double, dblFV
As Double, iType As Long) As Double
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X

	Available
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns the amount of an annuity based on constant and periodic payments, and at a constant interest rate.

Input parameters

- `dblRate`: This parameter indicates the interest rate per date of payment. For example, the rate per date of payment for a loan with a 6% annual interest rate, paid back by monthly dates of payment, would be:

```
0.06/12=0.005 or 0.5%
```

- `iNper`: This parameter contains the total number of dates of payment for the financial operation.
- `dblPV`: This parameter contains the actual value (or overall sum) for a series of payments to be made in the future.
- `dblFV`: This parameter contains the future value or the balance that you want to obtain after having paid the final date of payment. In general, and particularly when reimbursing a loan, this parameter is set to "0". In effect, once you have made all the dates of payment, the value of the loan is nil.
- `iType`: This parameter indicates the payment deadline. It can have one of the following values:
  - *0* if the payments are due in arrears (i.e. at the end of the period)
  - *1* if the payments are due in advance (i.e. at the start of the period)

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Notes

- The *`Rate`* and *`Nper`* parameters must be calculated using payments expressed in the same units.
- Amounts paid (expressed in particular by the *`Pmt`* parameter) are represented by negative numbers. Sums received are represented by positive numbers.

PPMT()

Internal BASIC syntax

```
Function PPMT(dblRate As Double, iPer As Long, iNper As Long, dblPV As
Double, dblFV As Double, iType As Long) As Double
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X

	Available
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns the amount of capital reimbursed for a given date of payment in an annuity based on constant and periodic payments and at a constant interest rate.

Input parameters

- `dblRate`: This parameter indicates the interest rate per date of payment. For example, the rate per date of payment for a loan with a 6% annual interest rate, paid back by monthly dates of payment, would be:

```
0.06/12=0.005 or 0.5%
```

- `iPer`: This parameter indicates the period for the calculation, between 1 and the value of the `Nper` parameter.
- `iNper`: This parameter contains the total number of dates of payment for the financial operation.
- `dblPV`: This parameter contains the actual value (or overall sum) for a series of payments to be made in the future.
- `dblFV`: This parameter contains the future value or the balance that you want to obtain after having paid the final date of payment. In general, and particularly when reimbursing a loan, this parameter is set to "0". In effect, once you have made all the dates of payment, the value of the loan is nil.
- `iType`: This parameter indicates the payment deadline. It can have one of the following values:
  - *0* if the payments are due in arrears (i.e. at the end of the period)
  - *1* if the payments are due in advance (i.e. at the start of the period)

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Notes

- The *`Rate`* and *`Nper`* parameters must be calculated using payments expressed in the same units.
- Amounts paid (expressed in particular by the *`Pmt`* parameter) are represented by negative numbers. Sums received are represented by positive numbers.

PV()

Internal BASIC syntax

```
Function PV(dblRate As Double, iNper As Long, dblPmt As Double, dblFV
As Double, iType As Long) As Double
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X

	Available
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns the actual amount of an annuity based on constant and periodic future deadlines, and on a fixed interest rate.

Input parameters

- `dblRate`: This parameter indicates the interest rate per date of payment. For example, the rate per date of payment for a loan with a 6% annual interest rate, paid back by monthly dates of payment, would be:

```
0.06/12=0.005 or 0.5%
```

- `iNper`: This parameter contains the total number of dates of payment for the financial operation.
- `dblPmt`: This parameter indicates the amount of the payment made at each date of payment. The payment generally includes both principal and interest.
- `dblFV`: This parameter contains the future value or the balance that you want to obtain after having paid the final date of payment. In general, and particularly when reimbursing a loan, this parameter is set to "0". In effect, once you have made all the dates of payment, the value of the loan is nil.
- `iType`: This parameter indicates the payment deadline. It can have one of the following values:
  - *0* if the payments are due in arrears (i.e. at the end of the period)
  - *1* if the payments are due in advance (i.e. at the start of the period)

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Notes

- The *`Rate`* and *`Nper`* parameters must be calculated using payments expressed in the same units.
- Amounts paid (expressed in particular by the *`Pmt`* parameter) are represented by negative numbers. Sums received are represented by positive numbers.

Randomize()

Internal BASIC syntax

```
Function Randomize(lValue As Long)
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X

	Available
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Initializes the random number generator.

Input parameters

- *`lValue`*: Optional parameter used to initialize the random-number generator of the `Rnd` function by specifying a new initial value. If this parameter is omitted , the value returned by the system clock is used as the initial value.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

RATE()

Internal BASIC syntax

```
Function RATE(iNper As Long, dblPmt As Double, dblFV As Double, dblPV
As Double, iType As Long, dblGuess As Double) As Double
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns the interest rate per date of payment for an annuity.

Input parameters

- `iNper`: This parameter contains the total number of dates of payment for the financial operation.
- `dblPmt`: This parameter indicates the amount of the payment to be made at each date of payment. The payment generally includes both principal and interest.
- `dblFV`: This parameter contains the future value or the balance that you want to obtain after having paid the final date of payment. In general, and particularly when reimbursing a loan, this parameter is set to "0". In effect, once you have made all the dates of payment, the value of the loan is nil.
- `dblPV`: This parameter contains the actual value (or overall sum) for a series of payments to be made in the future.
- `iType`: This parameter indicates the payment deadline. It can have one of the following values:

- *0* if the payments are due in arrears (i.e. at the end of the period)
- *1* if the payments are due in advance (i.e. at the start of the period)

- *dblGuess*: This parameter contains the estimated value of the interest rate per date of payment.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Notes

- Amounts paid (expressed in particular by the *Pmt* parameter) are represented by negative numbers. Sums received are represented by positive numbers.
- This function performs its calculation using iterations, starting with the value assigned in the *Guess* parameter. If no result is found after 20 iterations, the function fails.

RemoveRows()

Internal BASIC syntax

```
Function RemoveRows(strList As String, strRowNames As String) As String
```

Field of application

Version: 3.5

	Available

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Performs a deletion in a list of lines identified by the `strRowNames` parameter.

This function is useful when processing "ListBox" control type values. Values from this type of control are represented as arrays as described below:

- The "|" character is used as the column separator.
- The "," character is used as the line separator.
- Each line ends with a unique idenifier at the right of the "=" sign.

Input parameters

- `strList`: Source string containing the values of a "ListBox" control to be processed.
- `strRowNames`: Identifiers of lines to be deleted. The identifiers are separated by commas.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.

- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Example

```
Dim MyStr
```

MyStr=RemoveRows("a1|a2=a0,b1|b2=b0", "a0,c0") 'Returns "b1|b2=b0"

Replace()

Internal BASIC syntax

```
Function Replace(strData As String, strOldPattern As String,
strNewPattern As String, bCaseSensitive As Long) As String
```

Field of application

Version: 3.5

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Replaces all occurrences of the *strOldPattern* parameter with the *strNewPattern* parameter inside the character string contained in the *strData* parameter. The search for the *strOldPattern* parameter can be made case-sensitive using the value of the *bCaseSensitive* parameter.

Input parameters

- *strData*: Character string containing the occurrences to be replaced.
- *strOldPattern*: Occurrence to find in the string contained in the *strData* parameter.
- *strNewPattern*: Text replacing each occurrence found.
- *bCaseSensitive*: Depending on this parameter, the search is case sensitive (=1) or not (=0).

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Example

```
Dim MyStr
```

MyStr=Replace("youmeyoumeyou", "you", "me",0) 'Returns "mememememe"

MyStr=Replace("youmeyoumeyou", "You", "me",1) 'Returns "youmeyoumeyou"

MyStr=Replace("youmeYoumeyou", "You", "me",1) 'Returns "youmememeyou"

Right()

Internal BASIC syntax

```
Function Right(strString As String, iNumber As Long) As String
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Returns the rights most iNumber characters of the string parameter.

Input parameters

- `strString`: Character string to process.
- `iNumber`: Number of characters to return.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Example

```
Sub Main ()
  Dim LWord, Msg, RWord, SpcPos, UsrInp    ' Declare variables.
  Msg = "Enter two words separated by a space."
  UsrInp = InputBox(Msg)    ' Get user input.
  print UsrInp
  SpcPos = InStr(1, UsrInp, " ")    ' Find space.
  If SpcPos Then
    LWord = Left(UsrInp, SpcPos - 1)    ' Get left word.
    print "LWord: ";  LWord
    RWord = Right(UsrInp, Len(UsrInp) - SpcPos)    ' Get right word.
    Msg = "The first word you entered is " & LWord
    Msg = Msg & "." & " The second word is "
    Msg = "The first word you entered is <" & LWord & ">"
    Msg = Msg & RWord & "."
  Else
    Msg = "You didn't enter two words."
  End If
  Print Msg    ' Display message.
  MidTest = Mid("Mid Word Test", 4, 5)
  Print MidTest
End Sub
```

RightPart()

Internal BASIC syntax

```
Function RightPart(strFrom As String, strSep As String, bCaseSensitive
As Long) As String
```

Field of application

Version: 3.5

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Extracts the portion of a string to the right of the separator specified in the `strSep` parameter.

The search for the separator is performed from right to left.

The search can be made case sensitive using the `bCaseSensitive` parameter.

Input parameters

- *strFrom*: Source string to be processed.
- *strSep*: Character used as separator in the source string.
- *bCaseSensitive*: Depending on this parameter, the search is case sensitive (=1) or not (=0).

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Example

These examples illustrate use of the `LeftPart`, `LeftPartFromRight`, `RightPart`, and `RightPartFromLeft` functions on the same string: "This_is_a_test":

```
LeftPart("This_is_a_test","_",0)
```

Returns "This".

```
LeftPartFromRight("This_is_a_test","_",0)
```

Returns "This_is_a".

```
RightPart("This_is_a_test","_",0)
```

Returns "test".

```
RightPartFromLeft("This_is_a_test","_",0)
```

Returns "is_a_test".

RightPartFromLeft()

Internal BASIC syntax

```
Function RightPartFromLeft(strFrom As String, strSep As String,
bCaseSensitive As Long) As String
```

Field of application

Version: 3.5

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Extracts the portion of a string to the right of the separator specified in the `strSep` parameter.

The search for the separator is performed from left to right.

The search can be made case sensitive using the `bCaseSensitive` parameter.

Input parameters

- *strFrom*: Source string to be processed.
- *strSep*: Character used as separator in the source string.
- *bCaseSensitive*: Depending on this parameter, the search is case sensitive (=1) or not (=0).

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Example

These examples illustrate use of the `LeftPart`, `LeftPartFromRight`, `RightPart`, and `RightPartFromLeft` functions on the same string: "This_is_a_test":

```
LeftPart("This_is_a_test","_",0)
```

Returns "This".

```
LeftPartFromRight("This_is_a_test","_",0)
```

Returns "This_is_a".

```
RightPart("This_is_a_test","_",0)
```

Returns "test".

```
RightPartFromLeft("This_is_a_test","_",0)
```

Returns "is_a_test".

RmDir()

Internal BASIC syntax

```
Function RmDir(strRmDirectory As String) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Removes an existing directory.

Input parameters

- `strRmDirectory`: Full path of the directory to be removed.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

Rnd()

Internal BASIC syntax

```
Function Rnd(dValue As Double) As Double
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Returns a value containing a random number.

Input parameters

- `dValue`: Optional parameter whose value defines the mode of execution of the function:
  - Less than zero: The same number is generated each time.
  - Greater than zero: Next random number in the series.
  - Equal to zero: Last random number generated.
  - Omitted: Next random number in the series.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Notes

Before calling this function, you must use the `Randomize` function, without parameters, to initialize the random number generator.

Example

```
Dim MyNumber
```

Randomize

MyNumber= Int((10*Rnd)+1) 'Returns a random value between 1 and 10.

RTrim()

Internal BASIC syntax

```
Function RTrim(strString As String) As String
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Removes all trailing spaces in a string.

Input parameters

- *strString*: String to process.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Example

```
' This example uses the LTrim and RTrim functions to strip
leading ' and trailing spaces, respectively, from a string
variable.
```

```
' It uses the Trim function alone to strip both types of spaces.
' LCase and UCase are also shown in this example as well as the use
' of nested function calls

Sub Main
    MyString = "  <-Trim->  "    ' Initialize string.
    TrimString = LTrim(MyString)    ' TrimString = "<-Trim->  ".
    Print "|" & TrimString & "|"
    TrimString = LCase(RTrim(MyString))    ' TrimString = "  <-trim->".
    Print "|" & TrimString & "|"
    TrimString = LTrim(RTrim(MyString))    ' TrimString = "<-Trim->".
    Print "|" & TrimString & "|"
    ' Using the Trim function alone achieves the same result.
    TrimString = UCase(Trim(MyString))    ' TrimString = "<-TRIM->".
```

```
    Print "|" & TrimString & "|"
End Sub
```

Second()

Internal BASIC syntax

```
Function Second(tmTime As Date) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Returns the number of seconds contained in the time expressed by the `tmTime` parameter.

Input parameters

- *tmTime*: Parameter in Date+Time format to be processed.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

Example

```
Dim MyStr
```

MyStr=Second(AmDate())'Returns the number of seconds elapsed in the current hour, for example "30" if the time is 15:45:30

SetSubList()

Internal BASIC syntax

```
Function SetSubList(strValues As String, strRows As String,
strRowFormat As String) As String
```

Field of application

Version: 3.5

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X

	Available
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Defines the values of a sublist for a "ListBox" control.

Input parameters

- `strValues`: Source string containing the values of a "ListBox" control to be processed.
- `strRows`: List of values to add to or replace the characters contained in the string in the `strValues` parameter. The values are separated by the "|" character. The lines that are processed are identified by their identifier, situated to the right of the "=" sign. Unknown lines are not processed.
- `strRowFormat`: Formatting instructions for the sublist. Instructions are separated by the "|" character. This parameter has the following characteristics:
  - "1" represents the information contained in the first column of the sublist.
  - "i-j" can be used to define a group of columns.
  - "-" takes all columns into account.

An unknown column does not return a value.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Example

```
Dim MyStr
```

MyStr=SubList("a1|a2|a3=a0,b1|b2|b3=b0,c1|c2|c3=c0", "A2|A1=a0, B2|B1=b0", "2|1") 'Returns "A1|A2|a3=a0,B1|B2|b3=b0,c1|c2|c3=c0"

MyStr=SubList("a1|a2|a3=a0,b1|b2|b3=b0,c1|c2|c3=c0", "Z2=*,B2=b0", "2") 'Returns "a1|Z2|a3=a0,b1|B2|b3=b0,c1|Z2|c3=c0"

MyStr=SubList("a1|a2|a3=a0,b1|b2|b3=b0,c1|c2|c3=c0", "B5|B6|B7=b0,C5|C6,C7=c0", "5-7") 'Returns "a1|a2|a3=a0,b1|b2|b3||B5|B6|B7=b0,c1|c2|c3||C5|C6|C7=c0"

MyStr=SubList("a1|a2|a3=a0,b1|b2|b3=b0,c1|c2|c3=c0", "B1|B2|B3|B4=b0", "-") 'Returns "a1|a2|a3=a0,B1|B2|B3|B4=b0,c1|c2|c3=c0"

MyStr=SubList("A|B|C,D|E|F", "X=*", "2") 'Returns "A|X|C,D|X|F"

Sgn()

Internal BASIC syntax

```
Function Sgn(dValue As Double) As Double
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X

	Available
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Returns a value indicating the sign of a number.

Input parameters

- `dValue`: Number whose sign you want know.

Output parameters

The function can return one of the following values:

- 1: The number is greater than zero.
- 0: The number is equal to zero
- -1: The number is less than zero.

Shell()

Internal BASIC syntax

```
Function Shell(strExec As String) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	X

	Available
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Launches an executable program.

Input parameters

- `strExec`: Full path of the executable to be launched.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Example

```
Dim MyId\nMyId=Shell("C:\WinNT\notepad.exe")
```

Sin()

Internal BASIC syntax

```
Function Sin(dValue As Double) As Double
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Returns the sine of an number that is expressed in radians.

Input parameters

- `dValue`: Number whose sine you want to know.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Space()

Internal BASIC syntax

```
Function Space(iSpace As Long) As String
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Creates a string including the number of spaces indicated by the *`iSpace`* parameter.

Input parameters

- *iSpace*: Number of spaces to be inserted into the string.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Notes

This function can be used to format strings or to delete date in fixed length strings.

Example

```
Dim MyString
```

' Returns a string of 10 spaces.

MyString = Space(10)

' Inserts 10 spaces between two strings.

MyString = "Space" & Space(10) & "inserted"

Sqr()

Internal BASIC syntax

```
Function Sqr(dValue As Double) As Double
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Returns the square root of a number.

Input parameters

- `dValue`: Number whose square root you want to know.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Str()

Internal BASIC syntax

```
Function Str(strValue As String) As String
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Converts a number to a string.

Input parameters

- `strValue`: Number to convert to a string.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

StrComp()

Internal BASIC syntax

```
Function StrComp(strString1 As String, strString2 As String,
iOptionCompare As Long) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Compares two strings.

Input parameters

- `strString1`: First string.
- `strString2`: Second string.
- `iOptionCompare`: Comparison type. This parameter can be set to "0" for a binary comparison, or "1" for a text comparison.

Output parameters

- -1: `strString1` is greater than `strString2`.
- 0: `strString1` is equal to `strString2`.
- 1: `strString1` is less than `strString2`.

String()

Internal BASIC syntax

```
Function String(iCount As Long, strString As String) As String
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

String returns a string consisting of the *strString* character repeated over and over *iCount* times.

Input parameters

- *iCount*: Number of occurrences of the character.
- *strString*: Character used to compose the string.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

SubList()

Internal BASIC syntax

```
Function SubList(strValues As String, strRows As String, strRowFormat
As String) As String
```

Field of application

Version: 3.5

	Available
Built-in function	X
AssetCenter APIs	

	Available
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Returns a sublist of a list of values contained in a string representing the values of a "ListBox" control.

Input parameters

- `strValues`: Source string containing the values of a "ListBox" control to be processed.
- `strRows`: Identifiers of lines to be included in the sublist. The identifiers are separated by commas. Certain wildcard characters can be used:
  - "*" includes all identifiers in the sublist.
  - An unknown identifier returns an empty value for the sublist.
- `strRowFormat`: Formatting instructions for the sublist. Instructions are separated by the "|" character. This parameter has the following characteristics:
  - "1" represents the information contained in the first column of the list from which we are extracting a sublist.
  - "0" represents the identifier of the line in the list from which we are extracting a sublist.
  - "*" represents the information contained in all the columns (except the line identifier).
  - An unknown column does not return a value.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Example

```
Dim MyStr
```

MyStr=SubList("a1|a2|a3=a0,b1|b2|b3=b0,c1|c2|c3=c0", "a0,b0,a0", "3|2|3") 'Returns "a3|a2|a3,b3|b2|b3,a3|a2|a3"

MyStr=SubList("a1|a2|a3=a0,b1|b2|b3=b0,c1|c2|c3=c0", "*", "*|0") 'Returns "a1|a2|a3|a0,b1|b2|b3|b0,c1|c2|c3|c0"

MyStr=SubList("a1|a2|a3=a0,b1|b2|b3=b0,c1|c2|c3=c0", "*", "*=0") 'Returns "a1|a2|a3=a0,b1|b2|b3=b0,c1|c2|c3=c0"

MyStr=SubList("a1|a2|a3=a0,b1|b2|b3=b0,c1|c2|c3=c0", "*", "999=0") 'Returns "=a0,=b0,=c0"

MyStr=SubList("a1|a2|a3=a0,b1|b2|b3=b0,c1|c2|c3=c0", "z0", "*=0") 'Returns ""

MyStr=SubList("a1|a2|a3=a0,b1|b2|b3=b0,c1|c2|c3=c0", "*", "=1") 'Returns "=a1,=b1,=c1"

MyStr=SubList("A|B|C,D|E|F", "*", "2=0") 'Returns "B,E"

Tan()

Internal BASIC syntax

```
Function Tan(dValue As Double) As Double
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Returns the tangent of a number expressed in radians.

Input parameters

- `dValue`: Number whose tangent you want to know.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Time()

Internal BASIC syntax

```
Function Time() As Date
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Returns the current time.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.

- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Timer()

Internal BASIC syntax

```
Function Timer() As Double
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Returns the number of seconds elapsed since 12:00 AM.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

TimeSerial()

Internal BASIC syntax

```
Function TimeSerial(iHour As Long, iMinute As Long, iSecond As Long) As
Date
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns a time formatted according to the iHour, iMinute and iSecond parameters.

Input parameters

- *iHour*: Hour.
- *iMinute*: Minutes.
- *iSecond*: Seconds.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Example

Each of these parameters can be set to a numeric expression representing a number of hours, minutes or seconds. Thus in the following example:

```
TimeSerial(12-8, -10, 0)
```

Returns the value:

```
3:50:00
```

When the value of a parameter is out of the expected range (i.e. 0-59 for minutes and seconds and 0-23 for hours), it is converted to the parameter the next up. Thus, if you enter "75" for the *iMinute* parameter, it will be interpreted as 1 hour and 15 minutes.

The following example:

```
TimeSerial (16, 50, 45)
```

Returns the value:

```
16:50:45
```

TimeValue()

Internal BASIC syntax

```
Function TimeValue(tmTime As Date) As Date
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

This function returns the time portion of a "Date+Time" value.

Input parameters

- `tmTime`: "Date+Time" format date.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Example

The following example:

```
TimeValue ("1999/09/24 15:00:00")
```

Returns the value:

```
15:00:00
```

Trim()

Internal BASIC syntax

```
Function Trim(strString As String) As String
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	

	Available
AssetCenter Web APIs	X

Description

Returns a copy a string with the leading and trailing spaces removed.

Input parameters

- `strString`: String to process.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Example

```
' This example uses the LTrim and RTrim functions to strip
leading ' and trailing spaces, respectively, from a string
variable.
```

' It uses the Trim function alone to strip both types of spaces.

' LCase and UCase are also shown in this example as well as the use

' of nested function calls

Sub Main

MyString = " <-Trim-> " ' Initialize string.

TrimString = LTrim(MyString) ' TrimString = "<-Trim-> ".

Print "|" & TrimString & "|"

TrimString = LCase(RTrim(MyString)) ' TrimString = " <-trim->".

```
    Print "|" & TrimString & "|"
    TrimString = LTrim(RTrim(MyString))    ' TrimString = "<-Trim->".
    Print "|" & TrimString & "|"
    ' Using the Trim function alone achieves the same result.
    TrimString = UCase(Trim(MyString))    ' TrimString = "<-TRIM->".
    Print "|" & TrimString & "|"
End Sub
```

UCase()

Internal BASIC syntax

```
Function UCase(strString As String) As String
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Returns a copy of a sting in which all lowercase characters are converted to uppercase.

Input parameters

- *strString*: Character string to convert to uppercase.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Example

```
' This example uses the LTrim and RTrim functions to strip
leading ' and trailing spaces, respectively, from a string
variable.

' It uses the Trim function alone to strip both types of spaces.

' LCase and UCase are also shown in this example as well as the use

' of nested function calls


Sub Main

   MyString = "  <-Trim->  "    ' Initialize string.

   TrimString = LTrim(MyString)    ' TrimString = "<-Trim->  ".

   Print "|" & TrimString & "|"

   TrimString = LCase(RTrim(MyString))    ' TrimString = "  <-trim->".

   Print "|" & TrimString & "|"

   TrimString = LTrim(RTrim(MyString))    ' TrimString = "<-Trim->".

   Print "|" & TrimString & "|"

   ' Using the Trim function alone achieves the same result.

   TrimString = UCase(Trim(MyString))    ' TrimString = "<-TRIM->".
```

```
    Print "|" & TrimString & "|"
End Sub
```

UnEscapeSeparators()

Internal BASIC syntax

```
Function UnEscapeSeparators(strSource As String, strEscChar As String)
As String
```

Field of application

Version: 3.5

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Deletes all the escape characters from a string.

Input parameters

- *strSource*: Character string to process.
- *strEscChar*: Escape character to be deleted.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Example

```
Dim MyStr
```

MyStr=UnEscapeSeparators("you\|me\|you\|", "\") 'Returns "you|me|you|"

Union()

Internal BASIC syntax

```
Function Union(strListOne As String, strListTwo As String, strSeparator
As String, strEscChar As String) As String
```

Field of application

Version: 3.5

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X

	Available
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Merges two strings delimited by separators. Duplicates are deleted.

Input parameters

- `strListOne`: First string.
- `strListTwo`: Second string.
- `strSeparator`: Separator used to delimit the elements contained in the strings.
- `strEscChar`: Escape character. If this character prefixes the separator, it will be ignored.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

Example

```
Dim MyStr
```

MyStr=Union("a1|a2,b1|b2", "a1|a3,b1|b2", ",", "\") 'Returns "a1|a2,b1|b2,a1|a3"

MyStr=Union("a1|a2,b1|b2", "a1|a3\,b1|b2", ",", "\") 'Returns "a1|a2,b1|b2,a1|a3\,b1|b2"

Val()

Internal BASIC syntax

```
Function Val(strString As String) As Double
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Converts a string representing a number to a double.

Input parameters

- `strString`: Character string to convert.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the `AmLastError` function (and optionally the `AmLastErrorMsg` function) to find out if an error occurred (and obtain its associated message).

WeekDay()

Internal BASIC syntax

```
Function WeekDay(tmDate As Date) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Returns the day of the week contained in the date expressed by the `tmDate` parameter.

Input parameters

- `tmDate`: Parameter in Date+Time format to be processed.

Output parameters

The number returned corresponds to a day of the week where "1" represents Sunday, "2" Tuesday, ..., "7" Saturday.

Year()

Internal BASIC syntax

```
Function Year(tmDate As Date) As Long
```

Field of application

Version: 3.00

	Available
Built-in function	X
AssetCenter APIs	
Configuration script of a field or link	X
Definition script of a feature parameter	X
"Script" type action	X
Wizard script	X
FINISH.DO script of a wizard	
AssetCenter Web APIs	X

Description

Returns the year contained in the value expressed by the `tmDate` parameter.

Input parameters

- `tmDate`: Parameter in Date+Time format to be processed.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

Peregrine
SYSTEMS®
The Infrastructure Management Company™